VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Detekce normálního/abnormálního chování řidiče

Detection of Normal/Abnormal Driver Behaviour

2020 Jan Svěží

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jan Svěží**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Detekce normálního/abnormálního chování řidiče
Detection of Normal/Abnormal Driver Behaviour

Jazyk vypracování: čeština

Zásady pro vypracování:

Detekce anomálií v chování řidiče je významnou úlohou s dopadem na bezpečnost a s dopadem do oblasti samořiditelných aut. Je-li detekována závažná anomálie (např. závažná zdravotní indispozice), může být řízení během nouzového zastavení přepnuto do automatického módu a přivolána pomoc.

V diplomové práci proveďte:
1. K detekci použijte příznaků odvozených z kostry člověka; kostru můžete detekovat pomocí software OpenPose.
2. Experimentujte s různými příznaky použitými k detekci.
3. Detekci anomálií proveďte s využitím neuronových sítí; můžete použít např. síť LSTM. (Využijte některé z dostupných implementací konkrétní zvolené sítě, např. TensorFlow)
4. Funkčnost realizovaného řešení dostatečně ověřte.
5. Řešení a dosažené výsledky popište v textové části práce.

Seznam doporučené odborné literatury:

[1] Zhe Cao, Tomas Simon, Shih-En Wei, Yaser Sheikh: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields, CVPR 2017
[2] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, Puneet Agarwal: Long Short Term Memory Networks forAnomaly Detection in Time Series, ESANN 2015

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Dr. Ing. Eduard Sojka**

Datum zadání: 01.09.2019
Datum odevzdání: 30.04.2020

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 24. dubna 2020 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 24. dubna 2020 .....

Rád bych na tomto místě poděkoval hlavně dvěma lidem, díky kterým práce vznikla a byla úspěšně dokončena. Nejprve chci poděkovat svému vedoucímu, doc. Dr. Ing. Eduardu Sojkovi, který mě k práci přijal i přes mé několikanásobné předchozí neúspěchy a dal mi další šanci. Pan docent mi byl velmi kvalitním vedoucím, obrovskou pomocí a mentorem. Druhým člověkem, kterému bych chtěl poděkovat, je moje manželka. To ona mě podpořila v rozhodování, zda studovat magisterskou etapu studia, i při naší velmi náročné rodinné situaci. Držela při mně ve chvílích, kdy už se zdálo, že nedostuduji, nebo studium vzdám, a celou tu dobu mi dodávala sílu a energii. Děkuji Vám oběma. A díky i všem ostatním, kterých se má práce dotýkala, předně svým dětem a rodině.

Abstrakt

Diplomová práce se zabývá úlohou detekce abnormálního chování řidiče automobilu. Výstupem práce je zkoumání, zda-li a do jaké míry je možné úspěšně detekovat anomálie v chování osoby, která řídí automobil. Zaměřil jsem se na využití rekurentních neuronových sítí, primárně LSTM (Long short-term memory) a GRU (Gated Recurrent Unit). Neuronové sítě jsem modifikoval různými způsoby, např. změnou příznaků, změnou délky vstupních a výstupních vektorů, či změnou vnitřní struktury sítě. Příznaky používané pro učení neuronových sítí jsou odvozeny z kostry člověka pomocí knihovny OpenPose. K práci je přiložen software vytvořený v programovacích jazycích C++ a Python. Pomocí jazyka C++ a použitím knihoven OpenCV a OpenPose implementuji práci s videozáznamem - extrahuji příznaky pro neuronové sítě a modifikuji původní záznam pro vyobrazení detekovaných anomálií. A pomocí jazyka Python s využitím platformy TensorFlow a knihovny Keras implementuji neuronové sítě - to zahrnuje predikci časové řady a samotnou detekci anomálií.

Klíčová slova: Detekce anomálií, hluboké neuronové sítě, rekurentní neuronové sítě, LSTM, GRU, OpenCV, OpenPose, autonomní automobily, bezpečnost řízení, strojové učení

Abstract

This diploma thesis focuses on detection of abnormal car driver behavior. The goal of the thesis is to investigate whether and to what extent it is possible to successfully detect anomalies in the behavior of a person driving a car. For these purposes, recurrent neural networks were used - primarily LSTM (Long short-term memory) - and modified in various ways (e.g. by changing features, changing the length of input and output vectors, and changing the internal network structure). Features used for neural network learning were based on human skeleton by using OpenPose library. Attached is a software created in programming languages C++ and Python. C++ libraries OpenCV and OpenPose were used for video-analysis – To extract features for the neural network and modify the original video to show detected anomalies. Python platform TensorFlow and library Keras were then used to implement neural networks for time series prediction and anomaly detection.

Keywords: Anomaly detection, deep neural network, recurrent neural network, LSTM, GRU, OpenCV, OpenPose, autonomous vehicles, driver safety, machine learning

Obsah

Seznam použitých zkratek a symbolů

ML	–	Strojové učení (Machine Learning)
NN, NS	–	Neuronová síť (Neural Network)
DNN	–	Hluboká neuronová síť (Deep Neural Network)
CNN	–	Konvoluční neuronová síť (Convolutional Neural Network)
RNN	–	Rekurentní neuronová síť (Recurrent Neural Network)
BPTT	–	Algoritmus zpětného šíření v čase (Backpropagation Through Time algorithm)
LSTM	–	Long Short-Term Memory
GRU	–	Hradlová rekurentní jednotka (Gated Recurrent Unit)
BRNN	–	Obousměrné rekurentní neuronové sítě (Bidirectional RNN)
CTRNN	–	Continuous-Time Recurrent Neural Network
OpenCV	–	Open Source Computer Vision
SW	–	Software
FPS	–	Počet snímků za vteřinu (Frames Per Second)
CPU	–	Centrální procesorová jednotka (procesor)
GPU	–	Grafický procesor
ESN	–	Echo State Network
SVM	–	Metoda podpůrných vektorů (Support Vector Machine)
GAN	–	Generative adversarial network
IoV	–	Internet vozidel (Internet of Vehicles)
VAE	–	Variační autoenkodér (Variational autoencoder)
MSE	–	Střední kvadratická chyba (Mean squared error)
SNR	–	Poměr mezi sílou signálu a nechtěným (bezvýznamným) šumem (Signal-to-Noise Ratio)

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvodní slovo

Technologie jdou neustále kupředu a dnešní doba nabízí mnohé z nich i běžnému uživateli. Fotoaparáty či videokamery se minimalizují, a s rozmachem sociálních sítí jsou obrazové záznamy nedílnou součástí našich životů. Velmi kvalitní zařízení již dávno není otázkou velkých, těžkých, neskladných a drahých fotoaparátů či videokamer. Zařízení nahrávající video ve 4k rozlišení či fotoaparát pořizující kvalitní snímky i ve 20MPx, má dnes hodně lidí přímo ve svých mobilních telefonech. V průmyslu se kvalitní obrazová dokumentace pořizuje již mnohá léta. Například kvůli bezpečnosti v objektech, nahrávání záznamu lékařů na operačním sále, či k prozkoumávání neznámých, nebo těžko dostupných míst. Vývoj se tedy staví před otázku „Co s tolika záznamy?“. Na poli obrazových dat vznikl obor Digitální zpracování a analýza obrazu. V soukromém sektoru má využití kupříkladu pro:

- biometrické zabezpečení různých zařízení: *Sken obličeje, očí, otisku ruky, atd.*
- detekci obličeje na sociálních sítích

V průmyslu se obraz zpracovává např.:

- z bezpečnostních důvodů: *Detekuje a analyzuje osoby vstupující na nějaké specifické místo.*
- kvůli sběru dat pro statistické účely: *Počítá frekvenci automobilů, které projely daným úsekem v určitém čase.*
- z důvodu optimalizace procesů: *Zkoumá konkrétní pracovní postupy a navrhuje jejich vylepšení.*
- z důvodu optimalizace ve smyslu uspoření času lidských zdrojů: *Mnohé úkony mohou být pro člověka obtížné, ba přímo nemožné. V takových situacích může pomoci digitální zpracování obrazu, např. hledání určitých vzorů, analýza virových receptorů, vyhledávání nádorových útvarů v lidském těle a další.*

V oblasti zpracování digitálně pořízených snímků zaznamenává velmi strmý vzestup automobilový průmysl. Primárně mluvím o autonomním řízení různých typů vozidel či strojů, a to jak o 5. stupni - plné automatizaci, tak o stupni prvním, kam patří třeba adaptivní tempomat. Velmi významnou roli hrají také úlohy s dopadem na bezpečnost jízdy - analýza chování řidiče automobilu. Zde mohu zmínit systémy pro detekci únavy a doporučení odpočinku, monitorování pozornosti, sledování psychického stavu řidiče, úrovně stresu atd.

Ve své diplomové práci se zabývám detekcí abnormálního chování osoby za volantem. Jak ale charakterizovat abnormální chování? Je to chování, které není typické pro (jednoho konkrétního) řidiče. Je nutné nejprve zmapovat jeho normální chování, co je pro něj typické, a teprve poté je možné určit anomálie. V ideálním případě by bylo žádoucí takové abnormality klasifikovat. Pokud bychom byli schopni určit, o jaký problém se jedná, nejen že bychom mohli během nouzového zastavení přepnout do režimu autonomního řízení a zavolat záchrannou službu, ale mohli

bychom v některých případech podat rovnou první pomoc - např. vpuštění kyslíku do kabiny při hypoxii, či vpich kapsle adrenalinu do stehna při anafylaktickém šoku. Klasifikací se ve své diplomové práci nezabývám, jedná se však o vhodný námět na pozdější rozšiřování úlohy.

Diplomovou práci lze rozdělit do několika částí:

1. V první části představuji zadání a ukazuji jiná řešení úloh podobného charakteru.
2. Ve druhé pasáži popisuji mnou zvolené řešení a podávám přehled technologií, jež jsem v rámci diplomové práce používal.
3. Třetí část diplomové práce se věnuji vysvětlení nezbytně nutné teorie.
4. Ve čtvrtém úseku ukazuji a srovnávám výsledky jednotlivých experimentů.
5. A v závěru své práce hodnotím dosažené výsledky, úspěšnost řešení a navrhuji další možné rozšiřování práce.

2 Popis zadání úlohy

V závěru úvodu jsem letmo popsal problematiku, kterou se budu ve své diplomové práci zabývat. V této kapitole zadání rozvedu a popíši přesněji.

2.1 Motivace

Hrozby:

- vylitá káva
- popel od cigarety na holých nohách
- nepříjemné píchnutí hmyzu
- bláto na předním okně od kamionu jedoucího před autem
- náhle strnutí šíjového svalstva při špatném pohybu hlavy

Nebo také projevy některých závažných chorob, např.:

- svalová křeč
- prudké záškuby až nekontrolované pohyby končetin
- zhoršení dýchání
- ztráta vědomí
- panická ataka (záchvat)

Tyto a mnohé další problémy mohou nastat řidiči při jízdě automobilem. V takové chvíli se jeho tělo ve většině případů dostane do netypického stavu a polohy, a naruší tak jeho běžné chování při řízení. Stav tohoto typu můžeme nazvat abnormální či *anomálie*. Dojde-li k výrazné anomálii v chování řidiče za volantem jedoucího automobilu, může to způsobit velmi závažné komplikace s fatálními následky. Proto je v době moderních technologií důležité zabývat se rozpoznáním takových stavů, a zareagovat vhodným způsobem v co nejkratším čase po jejich detekci. Automobily mohou anomálii signalizovat (opticky či zvukem) a ujistit se, že je řidič v pořádku např. formou manuálního vypnutí signalizace. Také mohou přepnout do automatického módu, pomocí autonomního řízení nouzově zastavit a následně přivolat záchrannou službu. V budoucnosti by mohly poskytnout i základní první pomoc.

2.2 Zadání

Úkolem mé diplomové práce, a tedy i mým, je zjistit, zda je možné softwarově detekovat anomálie v chování řidiče výše popsaného typu, a do jaké míry. Ve své práci se nebudu zabývat výkonností vestavěných (embedded) zařízení, na kterých by byl software reálně nasazen. K implementaci budu používat běžný osobní počítač. Použiji vlastní datové podklady, video nahrávky, které zaznamenám přímo při řízení automobilu. Anomálie budu detekovat pomocí neuronových sítí. Vyzkouším více typů příznaků odvozených z lidské kostry popisujících řidičův stav. Během řešení práce se seznámím s teorií ohledně vhodných typů neuronových sítí a souvisejících témat. Získám vhled a naučím se používat některé již hotové implementace NS (úkolem není neuronovou síť implementovat od základů). Budu experimentovat s různými typy konfigurací zvolených sítí. A nakonec podám srovnání výsledků několika experimentů v hodnotách i grafech a zhodnotím míru kvality detekce, případně podám návrhy na možné budoucí rozšiřování výzkumu.

3 State-of-the-art aneb jak k problému přistupují ostatní

Při procházení internetu a hledání vědeckých prací, průzkumů a reálných aplikací, které popisují řešení stejných či podobných problémů, jsem našel více výzkumů zabývajících se „chováním“ automobilů než chováním řidiče. Mnohé práce se zaměřují na detekování anomálií okolního provozu, zda jedou ostatní vozidla ve správných pruzích, jestli jedou plynule atd. Další práce [1] se zabývá detekcí anomálií vnitřního systému automobilu. Autoři berou vozidla jako „kyber-fyzický systém se senzory“ a hledají anomálie v komunikaci celého ekosystému pomocí tzv. Skrytého Markovova modelu (Hidden Markov model). V jiné práci [2] se výzkumníci zabývají souvisejícím problémem: detekcí anomálií na vozovce (rozpoznání děr v silnici a jiné). V této práci autoři využívají metodu dolování dat (data-mining). Několik článků se zabývá detekcí řidiče na základě způsobu jízdy. Tyto práce ale opět berou v potaz spíše způsob použití různých součástí automobilu (plynový pedál, brzdový pedál a další). Nejbližšími studiemi zabývajícími se obdobným problémem, kterým se budu zabývat já, jsou detektory únavy řidiče. Články [3] a [4] popisují implementaci detektorů únavy pomocí rekurentních a konvolučních neuronových sítí.

Bohužel však všechny předchozí práce řeší detekci z odlišného úhlu pohledu. Nenašel jsem žádnou práci, která by řešila chování řidiče vozidla na základě příznaků odvozených z kostry člověka. Proto jsem zkusil prohledat i jiné domény, které využívají právě takovýchto klíčových bodů. Rád bych zmínil dvě práce. První z nich [5] se zabývá analýzou polohy těla pacienta pro detekci anomálií v pacientově chování. Jedná se o diplomovou práci absolventa univerzity v San Diegu z roku 2019. Autor rozpoznává anomálie pomocí prahu vypočteného na základě měření Euklidovské vzdálenosti mezi manuálně definovanou polohou kloubů člověka a polohou kloubů predikovanou pomocí generativní neuronové sítě typu VAE (Variační autoenkodér). Druhá práce [6] detekuje abnormální aktivity obecného charakteru. Automatizovaný kamerový systém pořizuje videozáznam, ze kterého jsou získávány polohy kloubů jednotlivých osob. Pomocí LSTM a konvolučních neuronových sítí typu autoenkodér se detektor učí normálnímu chování. S využitím nabytých dovedností pak detekuje anomálie. Průměrná úspěšnost detektoru je 93%.

Mnou prozkoumané práce přistupují k realizaci úloh podobného typu různými způsoby. Většina z nich však využívá k detekci některou z technik strojového učení a analýzy časových řad. Přístupy typu SVM (metoda podpůrných vektorů) analyzují úlohu s nejhoršími výsledky, naopak algoritmy využívající umělých neuronových sítí (konvoluční, rekurentní, GAN aj.) dosahují výsledků nejlepších.

Na základě nabytých informací jsem usoudil, že mnou zvolený způsob řešení úlohy detekce anomálií v chování řidiče je adekvátní vůči moderním technologiím využívaným v posledních letech.

4 Zvolené řešení úlohy

Úvodem se nejprve zamyslím nad zevrubným postupem řešení úlohy. Již ze zadání je zřejmé, že analýzu nebude možné provést žádným triviálním matematickým či jiným aparátem. Nejprve je nutné vypozorovat obvyklé chování člověka - jinými slovy naučit se, jak se lidská bytost chová v dané situaci. Úkol tedy zjevně spadá do problematiky strojového učení. Situaci „normální řízení“ není možné popsat exaktně, a přesto je potřeba porovnat stav řidiče v čase t se stavem řidiče v čase $(t+1)$. Je nezbytné nalézt v chování vzor, a v případě, že je tento vzor významně narušen, můžeme prohlásit, že se jedná o hledanou anomálii. Potřebujeme zvolit vhodnou reprezentaci stavu, a to takovou, která bude dostatečně relevantní. Stav bude reprezentován více hodnotami než jednou či dvěma. Bude se jednat o vektor zástupných hodnot, nejlépe čísel. Tento vektor nazveme „příznaky“ nebo „vektor příznaků“. Předchozí indicie vedou k logickému závěru, že bude potřeba využít složitějšího výpočetního nástroje. Např. struktury simulující chování lidského mozku - umělé neuronové sítě. Ta má schopnost se adaptovat a učit. Z mé úvahy vyplývá, že popis aktuálního stavu není triviální. Je tedy zjevné, že vzor chování, který se má stroj naučit, bude rovněž složitý. Tzn. že bude ke zvážení využití DNN, tzv. hlubokých neuronových sítí. DNN se vyznačují tím, že využívají velké množství vnitřních vrstev, a jsou tak schopny se naučit i velmi komplikované závislosti.

4.1 Obecný popis

Na konci úvodní části jsem se obecně zamyslel nad problémem a vydedukoval určité závěry. Na těchto základních myšlenkách jsem postavil své řešení, které je možné shrnout v několika bodech:

1. Pořízení videozáznamu

Na počátku vyvstává otázka, zda budu pro detekci anomálií potřebovat 3D rekonstrukci záznamu, a nebo bude plně dostačující dvourozměrný obraz. Vzhledem k tomu, že mým úkolem je vyhodnotit, zda je možné anomálie detekovat, dává smysl začít jednodušší variantou, a případně rozšiřovat.

2. Zpracování záznamu

Do tohoto bodu spadá volba příznaků, které mi dostatečně popíší aktuální stav, a jejich extrakce z obrazu. Stavem mám na mysli polohu řidiče (jak sedí, v jaké poloze má ruce, nebo sklon hlavy). Mám-li analyzovat anomálie v chování, tak zde nebudou hrát roli detaily, jako kam se řidič dívá, jeho mimika, ani jestli usíná nebo zda-li se potí. Bude nás zajímat, když náhle pustí ruce z volantu, začne sebou nezvykle házet, nebo dělat jiné neobvyklé pohyby. K zamyšlení se nabízí, zda by bylo vhodné všechny tyto parametry zkombinovat a zajistit tak dokonalejší výsledky. Jedná se o vhodný podnět pro rozšíření práce. Nyní se budu zabývat pouze anomáliemi způsobenými nezvyklou polohou či pohyby řidiče. Pro určení jeho polohy nám stačí několik málo klíčových bodů na těle (klouby horní části

těla, poloha krku, případně nosu). Další otázkou je, zda bude dostačovat samotná poloha kloubů, nebo by bylo vhodnější vzájemně je propojit. V experimentech zkoumám obě možnosti. Jako první jsem zvolil samotnou polohu kloubů. Poté jsem klouby propojil a zkoumám úhly mezi jednotlivými spoji. Například úhly mezi:

- ramenním kloubem, loktem a zápěstím
- jednou a druhou rukou vzájemně
- ramenními klouby a nosem

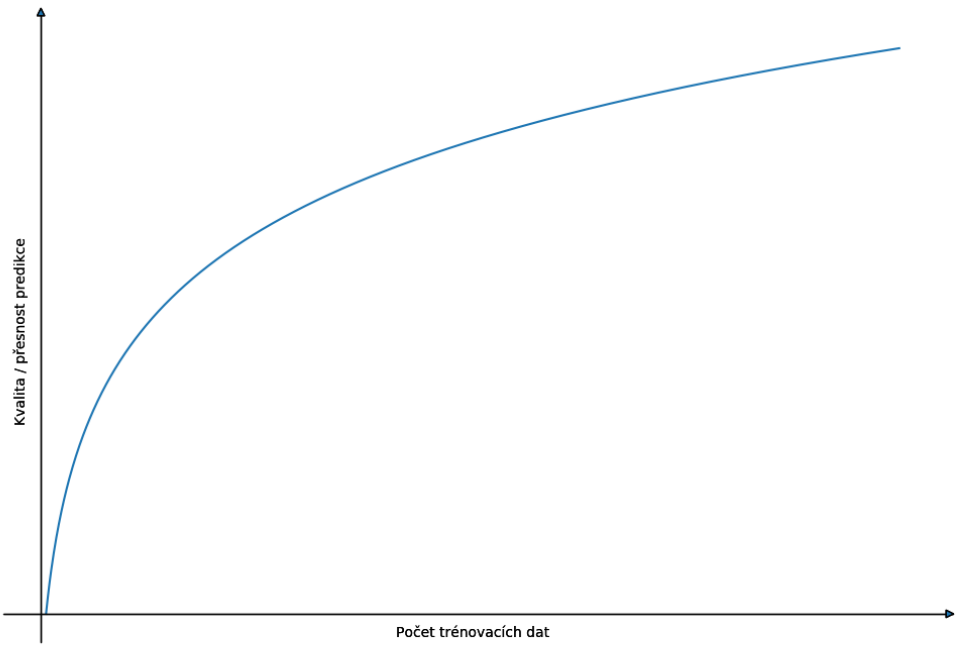
Pro získání souřadnic kloubů jsem použil volně šiřitelný nástroj OpenPose postavený na knihovně OpenCV společně s programovacím jazykem C++.

3. Volba modelu strojového učení

Existuje více metod strojového učení, jako např. diskriminační analýza, metoda podpůrných vektorů, Bayesovské sítě a další. Jak jsem popsal dříve, řešení úlohy detekce anomálií v lidském chování je komplikované. Nicméně lidský mozek je schopen takové situace rozpoznat poměrně snadno. Vhodným nástrojem se tedy jeví použití aparátu simulujícího funkčnost lidského mozku. Proto jsem pro řešení úlohy využil umělých neuronových sítí. A vzhledem k faktu, že je nutné zkoumat chování v čase, implementoval jsem tzv. rekurentní neuronové sítě (RNN). Jejich specifickým rysem je, že svůj výstup předávají dalším vrstvám a díky tomu zachovávají tzv. „časový kontext“. To znamená, že síť reflektuje celou historii předchozích poznatků. Takový proces je mnohem bližší lidskému mozku, než u běžných (nerekurentních) neuronových sítí. Pro implementaci neuronových sítí jsem využil volně dostupný framework Tensorflow a knihovnu pro hluboké učení Keras, oba nástroje pro programovací jazyk Python.

4. Proces učení

Čtvrtým krokem je samotný proces učení neuronových sítí. Využívám tzv. supervizovaného učení, či jinak, učení s učitelem. To znamená, že potřebuji data, na kterých budu schopen síť naučit normálnímu chování řidiče. Množství dat ovlivňuje kvalitu učení. Stejně jako u lidského mozku i zde platí pravidlo, čím více trénovacích dat má síť k dispozici, tím lépe je schopna situaci rozpoznat. Také u strojového učení platí průměrná učící křivka vzhledem k počtu trénovacích dat:



Obrázek 1: Průměrná učicí křivka [7]

Již několik hodin jízdy autem by mělo poskytnout vhodnou učící databázi:

$$1h \times 3600s \times cca\ 24\ FPS = 86400\ \textit{trénovacích dat za hodinu}$$

Jelikož se jedná o učení s učitelem, uživatel musí označit data, která vykazovala normální chování a vytvořit tak databázi trénovacích záznamů. Z hlediska učení je nejdůležitější typ a struktura neuronové sítě. Využil jsem rekurentních neuronových sítí (RNN), konkrétně nejpoužívanější sítě tohoto typu LSTM (Long short-term memory) a GRU (Gated Recurrent Unit) v různých konfiguracích. K detailnějšímu popisu se dostanu v dalších kapitolách.

5. Predikce normálního chování

Nyní mám vytvořený „umělý mozek“, který se naučil, jak se řidič za volantem chová normálně (obvyklým způsobem). Může se zdát, že jsem od detekce anomálií stále na míle vzdálený, protože síť neumí rozpoznat abnormální chování - umí naopak to normální. Naštěstí stejně jako lidský mozek, umělá neuronová síť má schopnost nejen rozpoznat normální chování, ale umí si ho také „představit“, umí jej predikovat. A tato myšlenka / dovednost je k detekci anomálií klíčová. Požádám RNN, aby mi předpověděla nějaký časový úsek normálního chování řidiče, n snímků. Každý snímek bude reprezentovaný vektorem m příznaků.

$$features\_vector = \begin{bmatrix} \begin{bmatrix} feature_1^{frame_1} \\ feature_2^{frame_1} \\ \vdots \\ feature_m^{frame_1} \end{bmatrix} & \begin{bmatrix} feature_1^{frame_2} \\ feature_2^{frame_2} \\ \vdots \\ feature_m^{frame_2} \end{bmatrix} & \cdots & \begin{bmatrix} feature_1^{frame_n} \\ feature_2^{frame_n} \\ \vdots \\ feature_m^{frame_n} \end{bmatrix} \end{bmatrix} \quad (4.1)$$

6. Srovnání predikce a reality

Zbývá poslední krok detekce. Zda-li je, respektive bylo, chování abnormální, mohu zjistit až ve chvíli kdy nastalo, nikoliv dříve. Pro detekci mám k dispozici vektor snímků reprezentující reálné chování řidiče (každý snímek je sám vektorem příznaků). Pokud si od umělé neuronové sítě nechám předpovědět týž počet snímků, získám dvě matice k porovnání - matice R a P reprezentující realitu a predikci. Vzdálenost predikce od reality spočítám jako Frobeniovu maticovou normu [8] rozdílu $A = R - P$.

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} \quad (4.2)$$

4.2 Použité knihovny

4.2.1 OpenCV

Open Source Computer Vision (OpenCV) [9] je SW knihovna s otevřeným zdrojovým kódem určená pro počítačové vidění a strojové učení. Knihovna obsahuje přes 2500 optimalizovaných algoritmů. Pomocí knihovny OpenCV je možné např. rozpoznávat obličeje, identifikovat objekty, klasifikovat lidské činnosti z videozáznamu, sledovat pohyb objektu a mnoho dalšího. Ve své diplomové práci využívám knihovnu pro čtení videa snímek po snímku. Jelikož mám v automobilu kameru připevněnou vzhůru nohama, tak použitím OpenCV provádím rotaci obrazu. Nakonec také využívám knihovnu pro vizualizaci detektoru anomálií a počítadlo snímků.

```
#include <iostream>
#include <opencv2/opencv.hpp>

int main()
{
   // open video stream
   cv::VideoCapture cap("/path/to/my/video/record.mp4");

   // check if the file was opened successfully
```

```
    if (!cap.isOpened())
        return -1;

    // get FPS
    float fps = cap.get(cv::CAP\_PROP\_FPS);

    while (true) {
        // capture frame-by-frame
        cv::Mat frame;
        cap >> frame;

        // if the frame is empty, finish reading
        if (frame.empty())
            break;

        // show frame
        cv::imshow("Anomaly visualizer", frame);

        // wait FPS time to show next frame or until ESC key is pressed
        if ((char)cv::waitKey(1000.0 / fps) == 27) {
            break;
        }
    }

    return 0;
}
```

Výpis 1: Čtení videozáznamu snímek po snímku pomocí OpenCV v jazyce C++

4.2.2 OpenPose

OpenPose [10, 11] je knihovna, ve které jsou implementované algoritmy pro detekci klíčových bodů lidského těla. Využitím OpenPose jsme schopni rozpoznat celkově 135 bodů. Primárně se jedná o klouby. Je možné analyzovat více osob v reálném čase a získat tak v jednom obraze souřadnice klíčových bodů na těle, rukou, nohou a obličeji. Body jsou rozloženy následujícím způsobem:

- 25 bodů na těle a nohou, z toho 6 bodů na chodidle
- 2×21 bodů na rukou

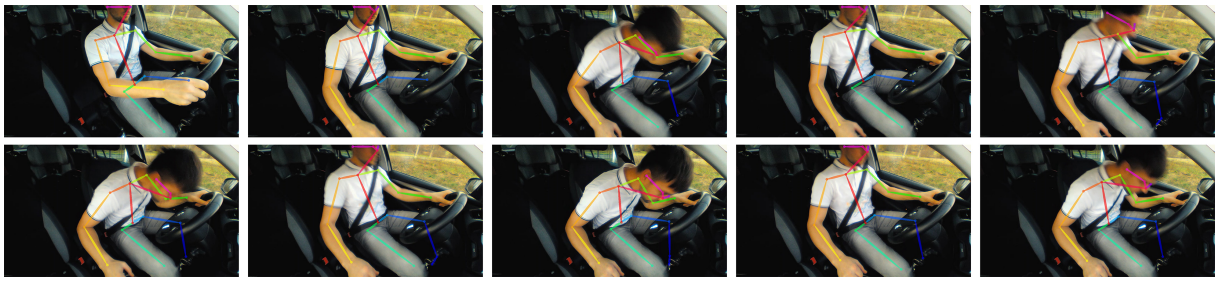
- 70 bodů na obličeji

Při analýze polohy řidiče vozidla stačí pouze podmnožina kloubů detekovaných pomocí knihovny OpenPose. Ve své diplomové práci využívám pouze následujících 11:

Hlava: nos, krk

Ruce: ramena, lokty, zápěstí

Tělo: levá strana boků, střed boků, pravá strana boků



Obrázek 2: Aplikace OpenPose při detekci klíčových bodů řidiče automobilu

4.2.3 TensorFlow

TensorFlow [12] je softwarová platforma s otevřeným zdrojovým kódem určená pro strojové učení. Má implementované velmi rozsáhlé portfolio SW nástrojů a knihoven. Společně tak poskytují opravdu silný a jednoduše uchopitelný nástroj pro aplikace založené na této oblasti umělé inteligence. Využitím TensorFlow je možné implementovat např.:

1. Různé druhy neuronových sítí
2. Generativní kontradiktorní sítě
3. Překládání textu využitím neuronových sítí
4. atd.

Platforma je využívána mnoha známými společnostmi, mezi něž patří např. Google, Intel, Twitter, Coca-Cola, AMD, NVidia, eBay, LinkedIn a další.

Ve své diplomové práci využívám TensorFlow společně s API knihovny Keras pro implementaci různých modelů neuronových sítí. Předně pak jednosměrnou i obousměrnou LSTM RNN.

4.2.4 Keras

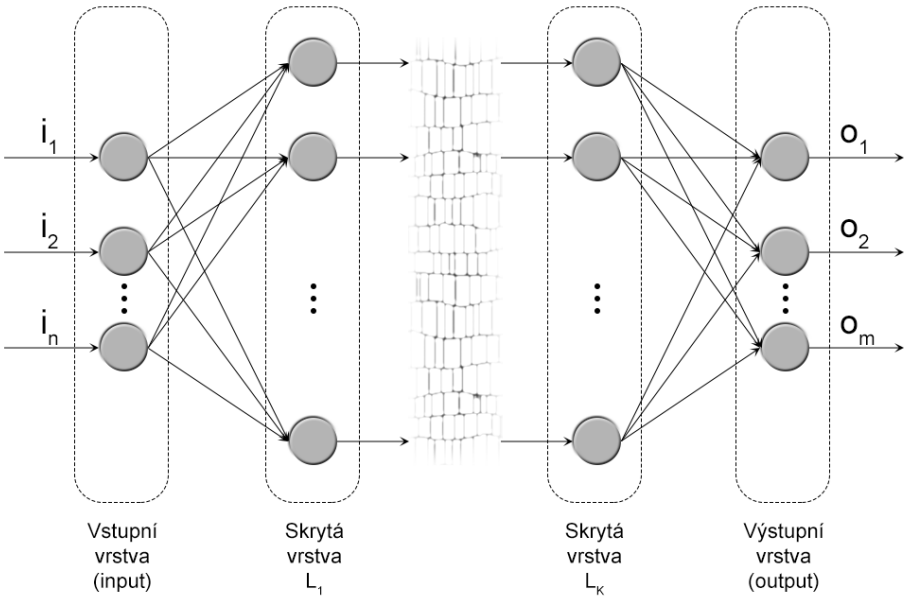
Keras [13] je vysokoúrovňové API pro implementaci neuronových sítí. Je napsané v programovacím jazyku Python. Spojení s API je v TensorFlow přímo implementované, a proto jej mohu v práci snadno využít. Pomocí Keras definuji typ modelu neuronové sítě a jednotlivé vrstvy. Keras podporuje konvoluční neuronové sítě (CNN), rekurentní neuronové sítě (RNN) a jejich kombinace a to jak na CPU, tak i s vysokou paralelizací na GPU.

5 Teorie

Úkolem mé diplomové práce není rozebírat teorii okolo neuronových sítí obecně. Vzhledem k faktu, že práce na neuronových sítích stojí, je ale žádoucí se určitou částí zabývat [14, 15, 16, 17, 18]. Umělé neuronové sítě fungující na principu učení s učitelem bychom mohli rozdělit na dva nejdůležitější typy: „dopředné (vícevrstvé) neuronové sítě“ (feedforward), někdy zvané také „vícevrstvý perceptron“, a „rekurentní neuronové sítě“ (feedback / recurrent).

5.1 Dopředné neuronové sítě

Základním principem dopředných NS je, že se signál přenáší pouze jedním směrem, od vstupních neuronů přes neurony vnitřních vrstev k neuronům vrstvy výstupní [obr. 3]. Signál nikdy nejde opačným směrem, ani se žádným způsobem nevrací do sítě. Pozor, tuto metodu není možné zaměňovat s metodou zpětné propagace (back-propagation). Zatímco dopředné NS jsou modelem neuronových sítí (popisuje postup výpočtu sítě), tak zpětná propagace je trénovací algoritmus (metoda pro adaptaci synaptických vah neuronů). Dopředné sítě se primárně používají pro klasifikaci či regresi.

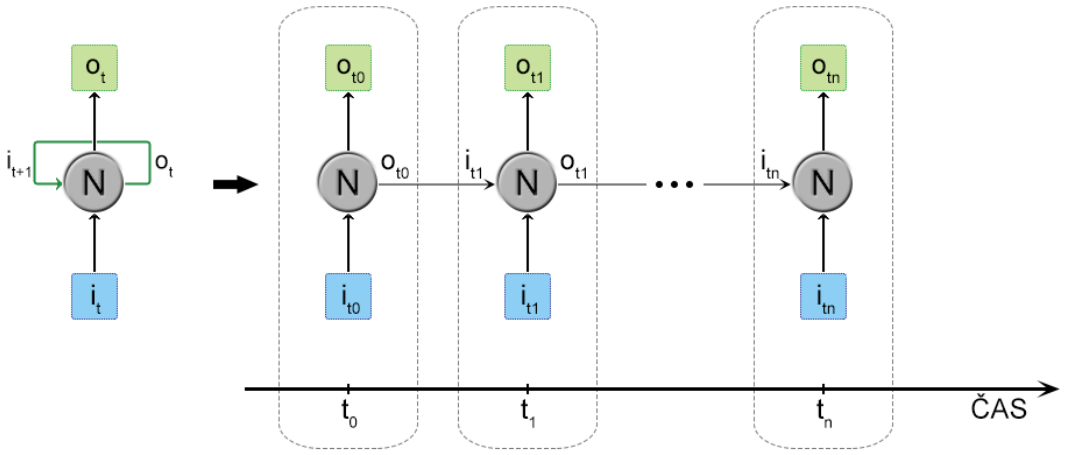


Obrázek 3: Vícevrstvá dopředná neuronová síť

5.2 Rekurentní neuronové sítě

Jelikož dopředné NS nemohou šířit signál zpět, nemohou se ani vzájemně ovlivňovat vnitřní výpočty. Síť má čistě reaktiví charakter. Žádný předchozí vstup neovlivňuje vstupy následující.

Síť není schopna brát v úvahu svoji historii (časový kontext). Někdy je ale tato vlastnost velmi žádoucí. Je to vlastně mnohem blíže lidskému uvažování. Lidé také nezačínají neustále od začátku. Je běžné, že např. rozvíjejícímu se učivu rozumíme víc a víc právě díky tomu, co jsme se naučili v předchozích lekcích. Nezačínáme pořád dokola, naše vědomosti zůstávají v paměti - mají perzistentní charakter. Časový kontext např. využijeme, chceme-li najít nějaké souvislosti či vzory. Typickými příklady takových úloh jsou např. generátory textu, jazykové překlady, predikce časových řad a další. Úlohy mají často společnou vlastnost - na vstupu bývají vzájemně se ovlivňující posloupnosti různých délek a na výstupu vyžadujeme opět libovolnou posloupnost. Aby bylo možné brát v potaz i zmíněný časový kontext, je potřeba ke vstupním informacím daného neuronu přidat i informace z předchozích průchodů. Za tímto účelem se do sítě přidávají tzv. rekurentní neurony, které zajišťují přenos stimulů z vyšších vrstev do vrstev nižších.



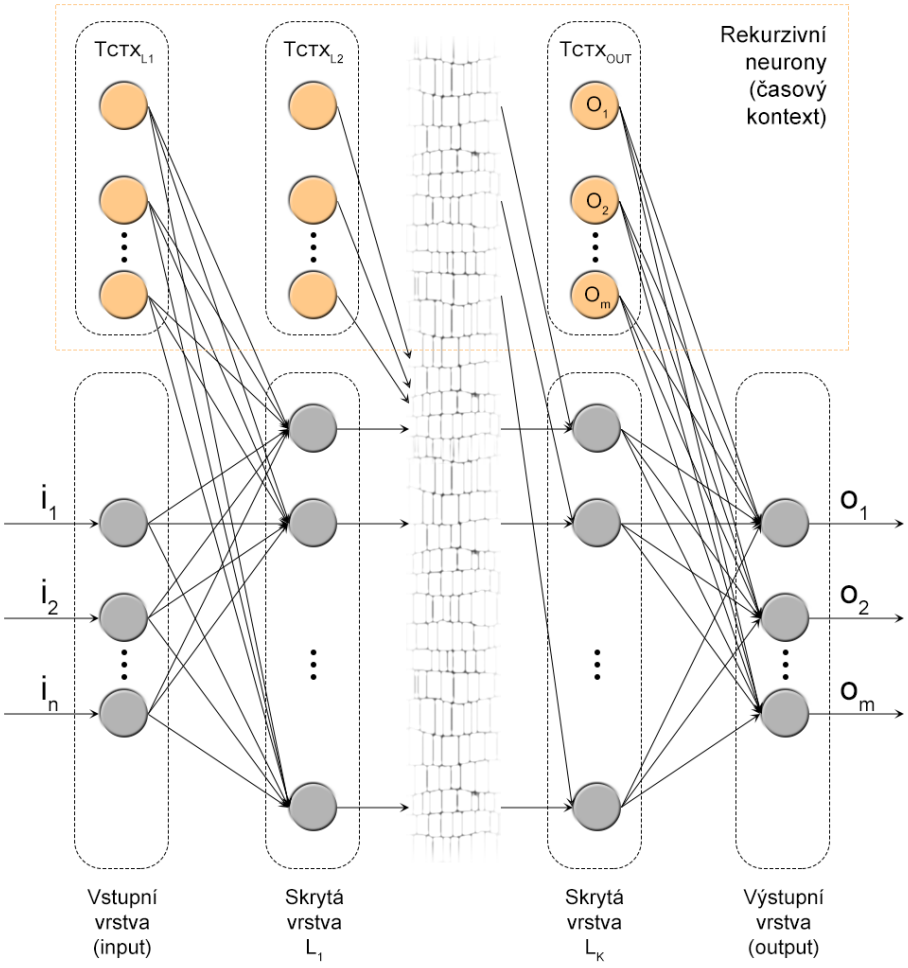
Obrázek 4: Princip RNN

Na obrázku [obr. 4] je znázorněn princip funkčnosti rekurentních neuronů, potažmo rekurentní (sub)sítě. V levé části obrázku je obecný schematický model. V čase t vstupuje signál i_t do neuronu N. Po zpracování signálu neuronem je na výstupu hodnota o_t, která slouží k dvěma účelům. Jednak je výstupní hodnotou (u vícevrstvé sítě by pak byla také vstupem do neuronů další vrstvy), a jednak se stává vstupním signálem do téhož neuronu N v čase $t + 1$. Na pravé straně obrázku je princip rekurentních neuronů rozvinutý. Jedná se stále o stejný neuron zpracovávající časovou řadu. Tzn. v čase t_0 přichází do neuronu N signál o síle i_{t0}. Neuron jej zpracuje a vrací hodnotu o_{t0}. V čase t_1 vstupuje do neuronu signál i_{t1}, a zároveň výstupní hodnota z času t_0, tj. o_{t0}. V čase t_2 je vstupem i_{t2} a o_{t1}, kde už je ale brán v potaz i signál o_{t0}. Platí tedy

následující ($S(x)$ je přenosová / aktivační funkce neuronu, např. sigmoida):

$$\begin{aligned}
o_{t0} &= S(i_{t0}w_{it0}) \\
o_{t1} &= S(i_{t1}w_{it1} + o_{t0}w_{ot0}) &&= S(i_{t1}w_{it1} + S(i_{t0}w_{it0})w_{ot0}) \\
o_{t2} &= S(i_{t2}w_{it2} + o_{t1}w_{ot1}) &&= S(i_{t2}w_{it2} + S(i_{t1}w_{it1} + S(i_{t0}w_{it0})w_{ot0})w_{ot1}) \\
&\vdots
\end{aligned}$$

Je tedy zjevné, že v každém čase bere model v potaz nejen aktuální stimul, ale i všechny předchozí „vnitřní stavy“. Tento model je možné analogicky rozšířit. Neuron N můžeme považovat za síť, a vstupy i_t za vektory hodnot. Rozšířený model pak tvoří vícevrstvou RNN, která je vyobrazená na obrázku [obr. 5].



Obrázek 5: Vícevrstvá RNN

Ve své diplomové práci mám za úkol detekovat anomálie chování. Je tedy nutné brát v úvahu, co řidič udělal „před chvílí“ v určitém časovém úseku. V opačném případě bych nebyl schopen

určit, zda se jedná o anomálii či nikoliv. Určité úkony obvykle následují za úkony jinými, např. když dám ruku z volantu pryč, pravděpodobně ji budu dávat na řadící páku nebo rádio. Jedná se tedy o analýzu časové řady a to typem učení „sequence-to-sequence“, tzn. že vstupem je posloupnost určité délky větší než 1 a výstupem je opět taková posloupnost (obecně jiné délky, než je délka vstupu). Z uvedených faktů je zřejmé, že pro analýzu abnormálního chování řidiče budu využívat RNN.

5.3 Trénování RNN

K adaptaci vah RNN se často používá algoritmus back-propagation, který je nejpoužívanějším trénovacím algoritmem i u dopředných neuronových sítí. Vzhledem k tomu, že RNN pracuje „v čase“, je nutné také trénovací algoritmus aplikovat se zachováním časového kontextu. V praxi to znamená duplikovat síť v každém časovém úseku/kroku. Dalo by se říci, že každý krok algoritmu back-propagation pro RNN je složen ze vstupu, jedné kopie sítě a výstupu. Chyba se pak počítá pro každý časový úsek a tvoří se jejich suma. Tato modifikace algoritmu back-propagation se nazývá Backpropagation Through Time (BPTT), neboli zpětné šíření v čase. Principem algoritmu back-propagation je minimalizace gradientu chybové funkce (loss funkce) se započítáváním aktuálních synaptických vah. Tzn. že se spočítá gradient chybové funkce a násobí se synaptickou váhou.

Zde ale může vzniknout problém u rekurentních neuronů. V případě, že bude váha rekurentního neuronu větší než 1, gradient vlivem opakovaného násobení stejnou hodnotou poroste exponenciální řadou, vlivem chybových signál pak začnou váhy oscilovat a učení sítě bude nestabilní. A naopak, pro hodnoty menší než 1 bude gradient exponenciálně klesat, chyba úplně zmizí a síť se přestane učit. [17] Je zjevné, že v případě závislostí, které jsou jen „několik“ časových kroků od sebe, by síť fungovala dobře, nicméně závislosti, které jsou od sebe více vzdálené, budou vykazovat potíže. Dva právě popsané úkazy se nazývají „explodující“ a „mizející“ gradient. Oba problémy podněcují špatné natrénování sítě. Dají se řešit dvěma způsoby. [16] Prvním možným řešením je vynechání rekurentních neuronů z trénovacího procesu úplně. K tomu slouží neuronové sítě typu ESN (Echo State Network), kterými se tu dále nebudu zabývat. Druhá možnost řešení zkreslujících gradientů je nastavit váhy rekurentních neuronů na konstantní hodnotu 1. Abychom ale v tomto případě získali informace o časovém kontextu co nejrelevantnější, je potřeba daný vnitřní stav vyřešit jinak. K tomuto účelu mohou sloužit například LSTM neuronové sítě.

5.4 Long Short-Term Memory (LSTM)

V publikaci [17] jsou podrobně popsané problémy explodujícího a mizejícího gradientu metody BPTT a jejich úspěšné řešení. V principu jde o docílení „konstantního toku chyby“ při zachování

časového kontextu rekurentních neuronů. Začnu s představou pro jednu „jednotku“ i s jediným spojem na sebe sama w_{ii}. Z BPTT plyne, že chybový signál jednotky i v čase t je:

$$\vartheta_i(t) = f_i'(net_i(t))\vartheta_i(t+1)w_{ii} \tag{5.1}$$

Konstantního toku chyby přes jednotku i dosáhneme, bude-li platit:

$$f_i'(net_i(t))w_{ii} = 1 \tag{5.2}$$

Integrací této rovnice zjistíme, že f_i musí být lineární a aktivační funkce jednotky i musí zůstat konstantní. Pomocí experimentů bylo prokázáno, že vyhovující je funkce identity s nastavením váhy na hodnotu 1:

$$\begin{aligned} \forall x : f_i(x) &= x \\ w_{ii} &= 1 \end{aligned} \tag{5.3}$$

Předchozí myšlenka ale naráží na dva problémy, které jsou způsobené naivní představou osamocené jednotky i. Reálně totiž jednotka nebude spojena sama se sebou, nýbrž s dalšími jednotkami. Prvním míněným problémem je „konflikt vstupních vah“ a druhým je „konflikt výstupních vah“.

Konflikt vstupních vah

Řekněme že na jednotku i přichází nový vstup j. Vzhledem k nutnosti adaptace je třeba jednotku i udržovat aktivní poměrně dlouhou dobu. Jelikož se vstupní váha používá pro uložení určitého vstupu a zároveň ignorování vstupů ostatních, w_{ij} často během aktivity dostává signály s konfliktními váhami.

Konflikt výstupních vah

Zde je situace obdobná, ovšem na výstupu. Dejme tomu, že z jednotky i odchází výstup k s váhou w_{ki}. Tato váha se opět používá ke dvěma účelům:

1. k načtení obsahu jednotky i v určitých časových krocích
2. k zabránění aby i narušovala k v jiných úsecích

Po celou dobu zpracování sekvence, kdy je i nenulová, dostává w_{ki} signály s konfliktními váhami.

Výše popsané problémy se vyskytují primárně u vzdálených časových úseků (samozřejmě mohou nastat i u blízkých). Jednotky je proto nutné z hlediska časového kontextu „optimalizovat“.

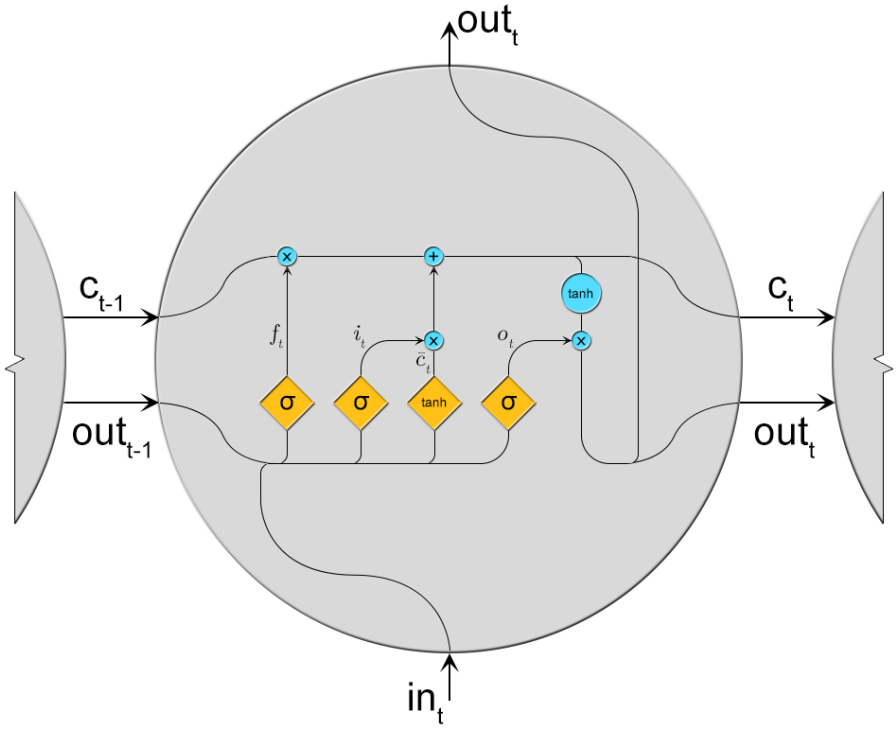
LSTM

Narozdíl od obvyklých RNN, kdy se v každém časovém kroku „kopíruje“ obvykle jedna vrstva neuronové sítě, např. typu tanh, LSTM kopíruje v čase složitější struktury. Tyto struktury obsahují čtyři vrstvy neuronové sítě, které jsou součástí tzv. „bran“. LSTM jednotky jsou, dá se říci, „paměťové buňky s bránami“. Tyto jednotky jsou založené na myšlence konstantního toku chyby s následujícími vylepšeními - paměťová buňka obsahuje:

1. Multiplikativní vstupní bránu
   chrání obsah paměti před chybami, které mohou vzniknout vlivem nerelevantních vstupů

2. Multiplikativní výstupní bránu
   chrání ostatní jednotky před chybami, které mohou vzniknout vlivem nerelevantního obsahu jiné buňky

Již takto optimalizované buňky zajišťují, že LSTM RNN jsou schopny řešit problémy, které vznikají explodujícím a mizejícím gradientem. Práce [17] však byla o tři roky později rozšířena o další bránu, tzv. „zapomínací bránu“ [18]. Úzkým hrdlem LSTM sítí bez zapomínací brány je fakt, že vnitřní stav jednotek může nekontrolovaně růst a způsobit tak pád neuronové sítě. Zapomínací brána je schopná ve správném čase resetovat vnitřní stav buňky a uvolnit tak vnitřní zdroje.



Obrázek 6: LSTM buňka s bránami [15]

Dále popíši základní princip průchodu LSTM buňkou [15, 16]. Veškeré poznatky se vztahují k [obr. 6] a výpočtům [5.4 - 5.9].

LSTM je schopná přidávat relevantní informace k aktuálnímu vnitřnímu stavu a odebírat nerelevantní. Veškeré signály jdou v buňce přes brány. Brány jsou tvořené neuronovou sítí a operátorem $\circ$ aplikujícím tzv. Hadamardův součin (násobení po složkách).

Vstupem LSTM buňky je $X_t = \{in_t, out_{t-1}\}$, kde in_t jsou nové vstupy a out_{t-1} jsou výstupy z předchozího kroku. Dále se přenáší také stav buňky z předchozího kroku c_{t-1}. Řekněme, že $w_{\{f,i,c,o\}}$ jsou váhy a $\theta_{\{f,i,c,o\}}$ jsou prahy pro brány a stav. σ a tanh jsou aktivační funkce.

S definicí předchozích pojmů můžeme spočítat vektory hodnot jednotlivých bran:

$$\text{zapomínací brána } f_t: \quad f_t = \sigma(w_f X_t + \theta_f) \tag{5.4}$$

$$\text{vstupní brána } i_t: \quad i_t = \sigma(w_i X_t + \theta_i) \tag{5.5}$$

$$\text{brána kandidátů } \bar{c}_t: \quad \bar{c}_t = \tanh(w_c X_t + \theta_c) \tag{5.6}$$

$$\text{nový stav buňky } c_t: \quad c_t = f_t \circ c_{t-1} + i_t \circ \bar{c}_t \tag{5.7}$$

$$\text{výstupní brána } o_t: \quad o_t = \sigma(w_o X_t + \theta_o) \tag{5.8}$$

$$\text{výstup buňky } out_t: \quad out_t = o_t \circ \tanh(c_t) \tag{5.9}$$

Co může síť zapomenout?

Po vstupu signálu do LSTM buňky je nejprve nutné rozhodnout, které informace je potřeba si nadále pamatovat aneb co zpracovávat v aktuálním stavu buňky. Vstupní vektor X_t jde tedy skrze zapomínací bránu f_t a pro každou hodnotu vektoru X_t vrátí pomocí sigmoidy číslo z intervalu $\langle 0; 1\rangle$. Pro hraniční bod 0 platí, že danou hodnotu vektoru je možné vypustit a pro 1 naopak, že aktuální hodnotu ponecháme v síti i nadále.

Nový stav

V síti již zůstaly pouze relevantní informace, se kterými se bude dále pracovat. Tyto je potřeba aktualizovat a případně k nim přidat nové hodnoty. K aktualizaci původních informací slouží tzv. vstupní brána, jenž je tvořena, stejně jako brána zapomínací, vrstvou sítě typu sigmoida. Další vrstva, tentokrát typu tanh, by se dala nazvat „brána kandidátů“. Jedná se o vektor nových potenciálních hodnot. Nyní jsme pomocí zapomínací brány zahodili nerelevantní vstupy $A = f_t \circ c_{t-1}$, připravili jsme si jejich aktualizaci a nové potenciální hodnoty $B = i_t \circ \bar{c}_t$. Nový vnitřní stav LSTM buňky tedy vytvoříme tím, že aktualizaci přičteme k původním hodnotám, tzn. $A + B$.

Výstup ze sítě

K poslednímu kroku výpočtu použijeme nový, aktualizovaný stav buňky. Tento vektor vpustíme do poslední brány jíž je brána výstupní. Pomocí sigmoidy rozhodneme, které hodnoty vnitřního stavu půjdou z buňky ven a přidáme k nim informace o vnitřním stavu rozšířené pomocí tanh do intervalu $\langle -1; 1\rangle$.

Jak je vidět, paměťová buňka si uchovává svůj vnitřní stav a důležité informace (vč. zpětné propagace chyby) jsou realizovány skrze něj. Proto mohou mít spoje mezi rekurentními neurony konstantní váhu s hodnotou 1. Jak jsem popsal výše, je zřejmé, že se tím LSTM RNN zbavila explodujících a mizejících gradientů a je schopná si zapamatovat mnohem delší úseky s časovými kontexty než běžné typy RNN.

5.5 Gated Recurrent Unit (GRU)

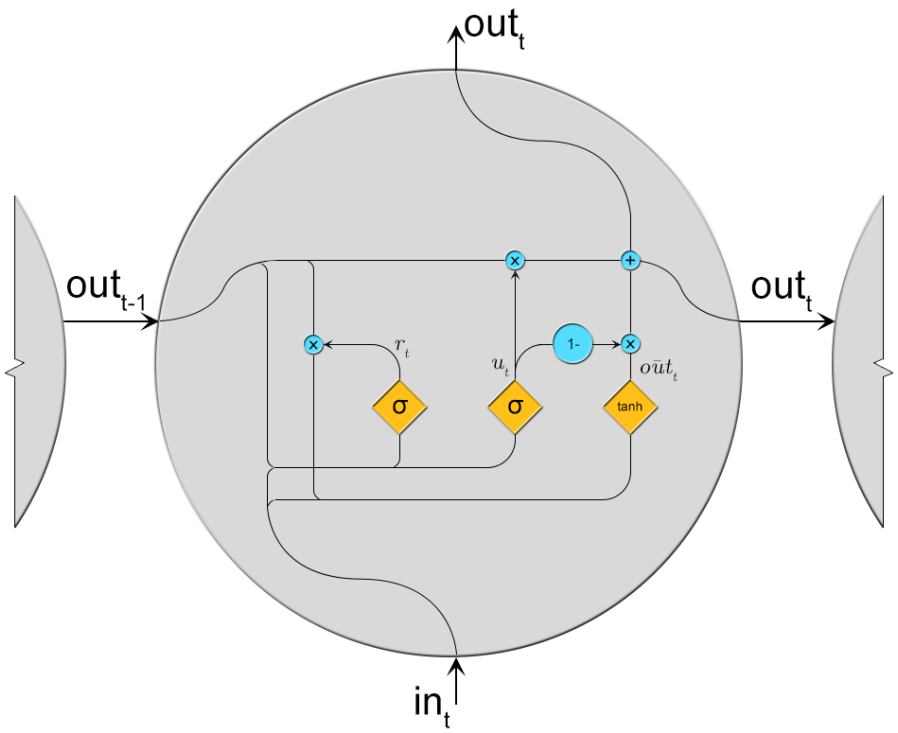
GRU je další variantou RNN, která řeší problémy explodujícího a mizejícího gradientu. Opět se jedná o síť reprezentovanou buňkami s bránami. Buňky GRU jsou podobné LSTM s tím rozdílem, že mají pouze dvě hlanvní brány:

Aktualizační brána (update gate)

Aktualizační brána je alternativou spojení zapomínací a vstupní brány LSTM buňky.

Resetovací brána (reset gate)

Resetovací brána slouží k rozhodnutí, jakou část informací z minulosti je možné zapomenout.



Obrázek 7: Gated Recurrent Unit (rekurentní jednotka s bránami) [15]

Princip průchodu GRU je podobný LSTM buňce, proto se ním nebudu zabývat nijak dopodrobna a popíši jej stručně [19]. Základní princip je vidět z obrázku [obr. 7] a výpočtů [5.11 - 5.13].

Jak je dáno principem RNN, buňka přijímá nové vstupy in_t a výstupy z předchozího kroku out_{t-1}, které zároveň nesou informaci o vnitřním stavu buňky. Proměnné $w_{\{u,r,c\}}$ a $v_{\{u,r,c\}}$ jsou váhy pro brány a stav. σ a tanh jsou aktivační funkce a operátor $\circ$ realizuje Hadamardův součin.

$$\text{resetovací brána } r_t: \quad r_t = \sigma(w_r in_t + v_r out_{t-1}) \tag{5.10}$$

$$\text{aktualizační brána } u_t: \quad u_t = \sigma(w_u in_t + v_u out_{t-1}) \tag{5.11}$$

$$\text{kandidát výstupu } \bar{out}_t: \quad \bar{out}_t = \tanh(w_{\bar{out}} in_t + r_t \circ u_{\bar{out}} out_{t-1}) \tag{5.12}$$

$$\text{výstup buňky } out_t: \quad out_t = (1 - u_t) \circ \bar{out}_t + u_t \circ out_{t-1} \tag{5.13}$$

Resetovací brána

Výpočet hodnot v resetovací bráně je stejný, jako u brány aktualizační. Rozdíl je ale v hodnotách vah a v místě použití brány. Tato brána slouží naopak k rozhodnutí, které z doposud relevantních informací je možné zapomenout.

Aktualizační brána

Určuje, které informace si bude síť dále pamatovat, a které může zapomenout. A stejně jako vstupní brána v LSTM slouží aktualizační brána k úpravě relevantních informací a k přidání nových podnětů. Vstupní vektory in_t a out_{t-1} jdou přes aktualizační bránu, která pomocí sigmoidy rozhodne, do jaké míry je nutné aktuální vstupy propagovat do budoucích kroků.

Modifikace obsahu buňky

S využitím Hadamardova součinu hodnot resetovací brány a vstupů z předchozího kroku výpočtu se do buňky uloží relevantní data z minulosti a aktuální nové vstupy. Získáme tak jakéhosi kandidáta na nový stav, který bude přeposlán do dalšího kroku výpočtu sítě.

Výstup ze sítě

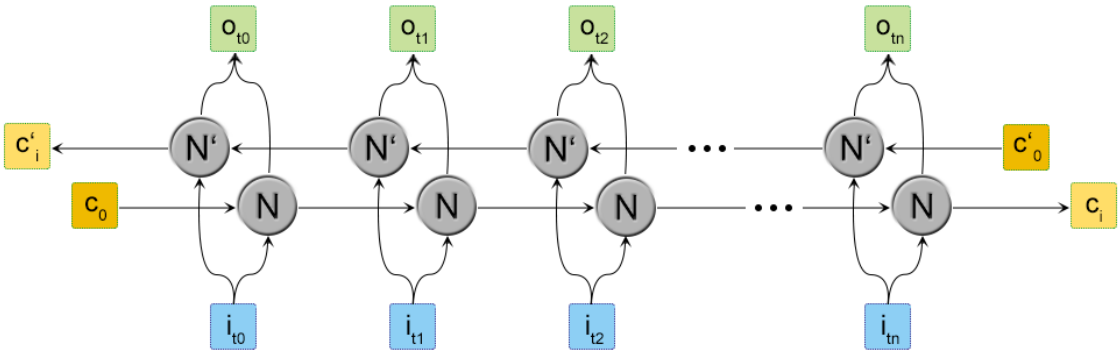
Pro výpočet nového vnitřního stavu buňky, a tedy i výstupu ze sítě, je nutné pomocí hodnot aktualizační brány upravit stav z předchozího kroku sítě a také stimuly aktuálního kandidáta na výstup. Výstupem je pak součet takto ovlivněných informací.

Je zjevné, že GRU je počítána pomocí menšího množství operací, a proto je při tréninku časově efektivnější. LSTM je obecně silnější sítí, nicméně na řešení většiny úloh podávají oba typy sítě obdobné výsledky. Je tedy často nutné vyzkoušet jak LSTM, tak i GRU a srovnat jejich výsledky.

5.6 Obousměrné rekurentní neuronové sítě (BRNN)

Pomocí běžných (jednosměrných) RNN jsme schopni pro předpověď výstupu y_{t_c} [20, 21] v určitém čase t_c pracovat se všemi dostupnými informacemi z minulosti až do úseku t_c (tj. $x_t, t = 1, 2, \cdots, t_c$). Avšak někdy bývá užitečné využít k předpovědi i ty informace, které přichází až po t_c. Pomocí RNN lze tyto informace získat zpožďováním výstupu o určitý (teoreticky libovolný) počet časových kroků M, ale prakticky začne síť po určité době vracet špatné výsledky. Nepříjemné na této vlastnosti je, že limit M není možné spočítat, lze jej zjistit pouze zkoušením.

Obousměrná RNN popsané omezení řeší. Hlavní myšlenkou je použití dvou na sobě nezávislých RNN pracujících se stejnými vstupy v opačném směru, tj. RNN A postupuje v „pozitivním směru toku času (dopředné stavy)“ a RNN B v „negativním směru toku času (zpětné stavy)“. Výstupy z A nejsou spojené se vstupy B, a ani naopak. Výstupy y_A a y_B jsou obvykle sjednoceny v každém kroku, a proto má síť v každém časovém rámci informace jak s časovým kontextem minulosti, tak budoucnosti. Princip je znázorněn na obrázku [obr. 8].



Obrázek 8: Princip obousměrných RNN [22]

5.6.1 Trénování BRNN

Jelikož jsou sítě A a B vzájemně nezávislé, je možné je v principu trénovat stejnými algoritmy jako běžné RNN. Je-li však pro trénování použit např. algoritmus BPTT, není možné stav a výstupní neurony na začátku a na konci trénovacích dat (v čase $t = 1$ pro dopředný směr a v $t = T$ pro zpětný směr) aktualizovat současně. K jejich nastavení se obvykle používá buď konkrétní hodnoty, např. $0,5$ nebo volba využitím samotného učícího procesu. Analogicky to rovněž platí pro informace o aktuálních časových kontextech. Tyto hodnoty bývají nastavené na 0. Obecně lze trénovací proces popsat ve třech krocích:

1. **Dopředný průchod:**
   Algoritmus projde nejprve dopředné stavy ($t = 1 \Rightarrow T$), pak zpětné stavy ($t = T \Rightarrow 1$) a nakonec výstupní neurony.

2. **Zpětný průchod:**
   Algoritmus nejprve prochází výstupní neurony, pak dopředné stavy ($t = 1 \Rightarrow T$) a nakonec zpětné stavy ($t = T \Rightarrow 1$)

3. **Aktualizace vah**

5.7 Zvolené příznaky

V obecném popisu řešení úlohy (v sekci 4.1) jsem poukazoval na data, která jsou stěžejní pro učící proces umělé neuronové sítě. Stejně tak jsou ale důležitá pro biologickou neuronovou síť. Oběma typům sítí stačí k popisu normálního chování několik málo údajů. Lidské neuronové síti ale vyhovují údaje na vyšší úrovni abstrakce, protože lidský mozek detaily zpracovává podvědomě. Člověku stačí k popisu normálního chování řidiče např. následující pozorování:

- řidič sedí
- je opřený o opěradlo
- ruce má na volantu, případně jednou pohybuje (řadí, zapíná rádio, nastavuje zrcátko atd.)
- hlavu má narovnanou tak, že mu přes přední sklo vidíme oči, nos i ústa
- hlava není nijak „neobvykle“ skloněná

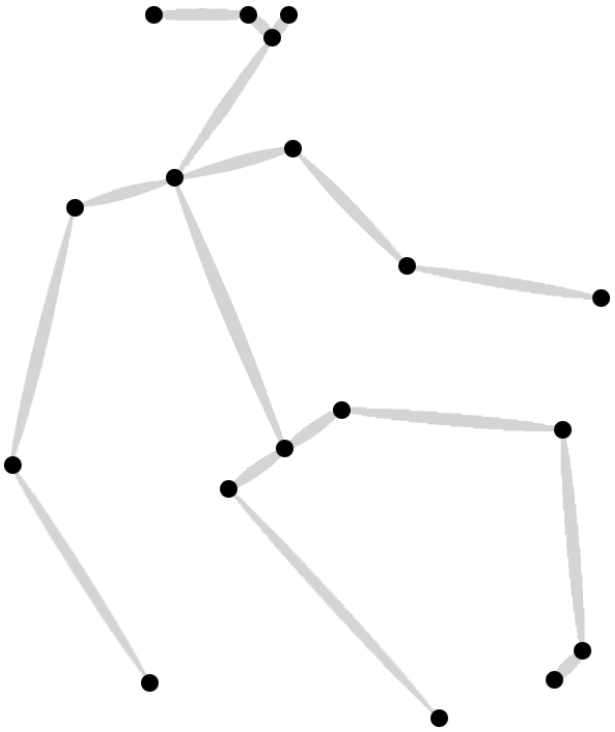
Tato pozorování jsou již poměrně komplexního charakteru, protože umělá neuronová síť neví, co znamená, když je řidič opřený o opěradlo. Obdobně je lidský mozek schopný vyhodnotit nezvyklé chování řidiče, v případě že:

- sebou rychle trhne
- se náhle prudce předkloní
- nečekaně pustí ruce z volantu a rozhodí je do stran
- uvede tělo do nezvyklé polohy či místa

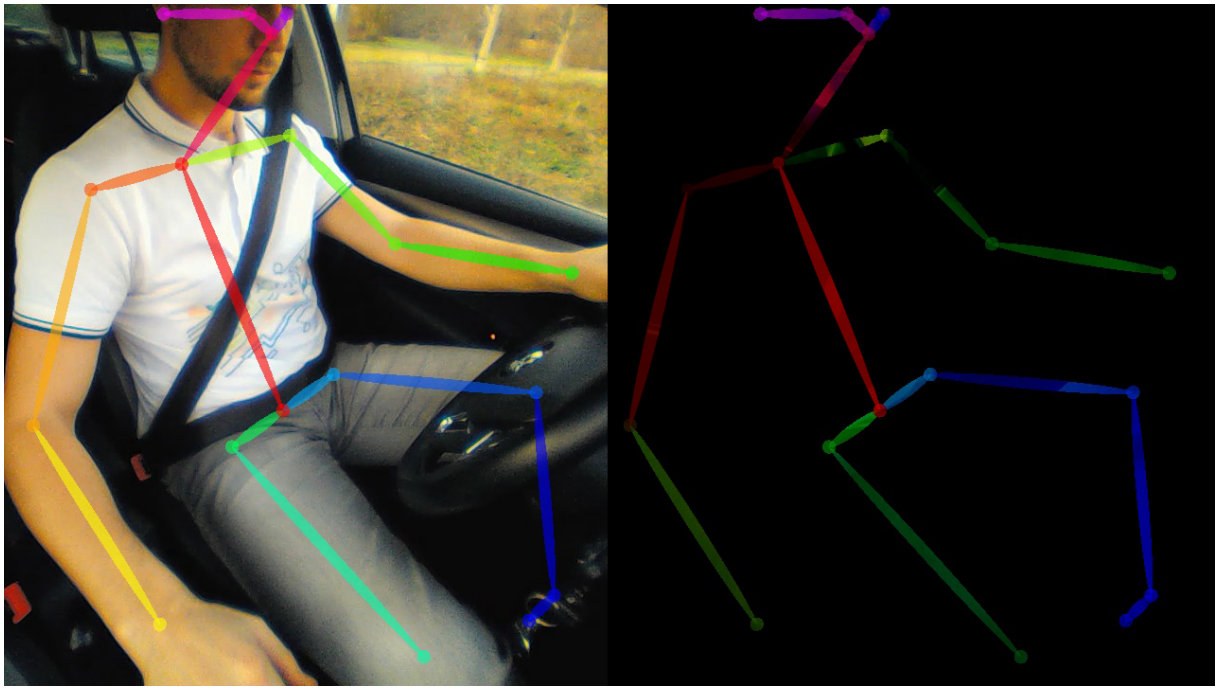
Lidskému oku stačí jeden takový vstupní signál a ono okamžitě spustí obrovské množství podvědomých procesů. Biologická neuronová síť se aktivuje a v mžiku vyhodnotí komplexní informaci jako abnormální stav.

Co ale model takové sítě? Jaké informace jsou potřeba pro detekci anomálií výše popsaného typu? Pokud si znovu přečteme příklady pozorování normálního a nezvyklého chování řidiče, zjistíme, že se všechna týkají polohy těla této osoby. Polohu těla je možné popsat pomocí relativně

snadno detekovatelných klíčových bodů. Těmito klíčovými body jsou klouby. Jestliže se umělá neuronová síť naučí, na jakém (přibližném) místě se nachází jednotlivé klouby, když osoba za volantem řídí, pak bude poměrně snadno schopná určit, do jaké míry se poloha odchyluje od normy. Popis pomocí polohy kloubů se mi jevil jako optimální, a proto jsem jej ve své práci využil. Pro detekci pozice kloubů jsem používal k tomu určený nástroj OpenPose (kap. 4.2.2) postavený na knihovně OpenCV (kap. 4.2.1).

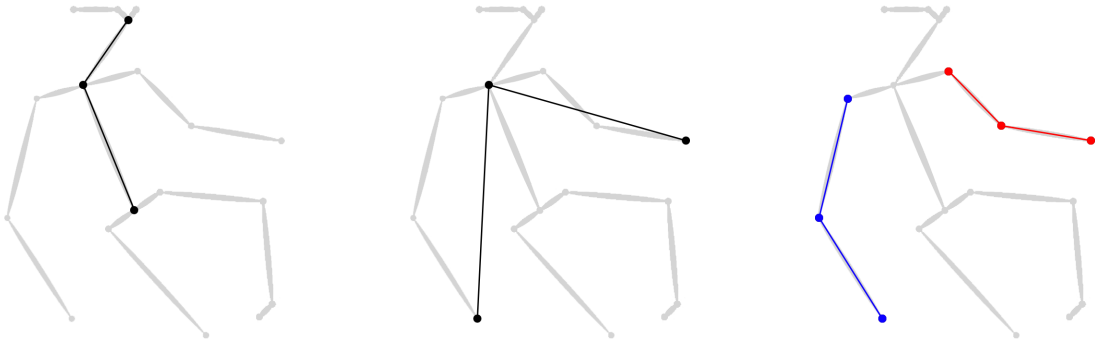


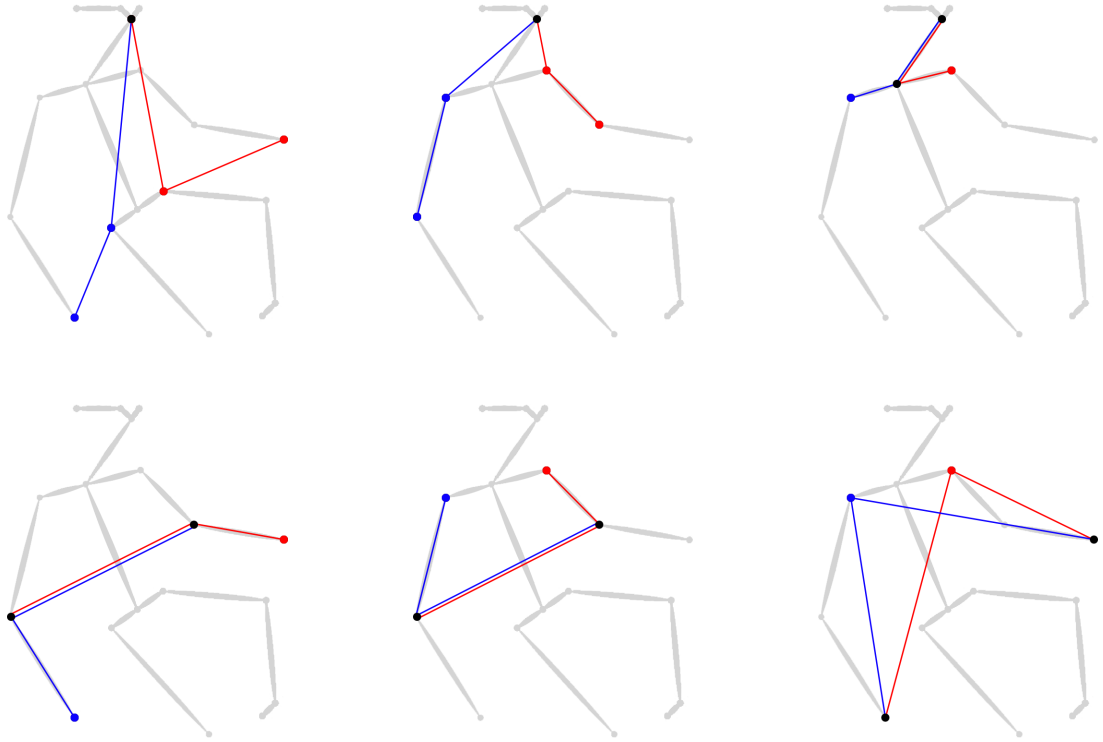
Obrázek 9: Detekované pozice kloubů pomocí knihovny OpenPose



Obrázek 10: Detekované pozice kloubů přímo v obraze

Při procesu učení jsem se zamýšlel, zda mohu použít pouhé souřadnice pozice jednotlivých kloubů v obraze, nebo zda-li bude nutná nějaká předchozí úprava. Rozhodl jsem se pro obojí. Po zamyšlení se nad problémem jsem si uvědomil, že pouhé souřadnice polohy kloubů by mohly být náchylné na relativně malé vnější změny, jako např. lehké natočení či posunutí kamery, nebo změna polohy celého posedu řidiče. Napadlo mě tedy, že lepší výsledky by mohly vykazovat úhly mezi jednotlivými částmi kostry. Při stejné aktivitě totiž úhly zůstávají velmi podobné i navzdory posunu kamery či jejímu pootočení, rozlišení snímaného obrazu a dalšími podněty. Úhly jsem volil dle vlastního úsudku, které z nich se mohou změnit při detekované anomálii. Zkoušel jsem několik obměn zvolených úhlů, ale vzhledem k faktu, že dosahovaly všechny velmi podobných výsledků, zůstal jsem pouze u původní sady. Všechny experimenty týkající se práce s úhly mezi jednotlivými klouby jsem prováděl právě na ní. Na obrázcích níže jsem množinu úhlů vyznačil do siluet s detekovanými klouby.





V sekci (6) jsou výsledky experimentů jak nad souřadnicemi pozic kloubů, tak nad příznaky definovanými jako úhly mezi specifickými klouby. Ve své diplomové práci jsem experimentoval s výše popsanými dvěma typy příznaků. Kdybych šel ještě o krok dál, bylo by více než vhodné posunout se do trojrozměrného prostoru a zkoumat úhly mezi jednotlivými částmi kostry ve 3-D. K detekci kloubů ve 3-D prostoru by bylo nutné paralelní nahrávání videozáznamu minimálně ze dvou kamer, rekonstrukce obrazu a detekce souřadnic kloubů ve 3-D. I k tomuto účelu by bylo možné použít knihovnu OpenPose. Není pro 3-D rekonstrukci a detekci přímo stavěná, ale v manuálu [10] je několik příkladů tohoto typu. Experimenty zahrnující trojrozměrnou detekci anomálií ponechávám na případné rozšiřování práce.

Neobvyklé situace nebo problémy bychom mohli popsat pomocí dalších rysů, např. z podrobné analýzy obličeje, či míry pocení řidiče aj. Nicméně, ve své práci se zabývám pouze výše popsanými typy anomálií.

5.8 Co od sítě očekávám?

V diplomové práci detekuji abnormální chování řidiče. Pracuji tedy s časovou řadou vektorů hodnot, které popisují aktuální stav řidiče. V každém časovém kroku předpovídám možné budoucí rámce a srovnávám je s realitou. Ve své práci jsem experimentoval se dvěma přístupy:

1. V prvním z nich dávám síti určitý časový úsek (pro ilustraci např. 3 sekundy) právě uběhnutého sledu událostí a očekávám, že mi předpoví jiný úsek (např. 1 sekundu) chování řidiče, které se jeví jako normální.

Tento přístup dává smysl, protože člověk za volantem jedná v určitých vzorech, postupech. Když řidič položí ruku na řadicí páku, po zařazení ji vrací zpět na volant - je zde daný sled událostí. I lidský mozek situaci vyhodnotí obdobně. Zeptáte-li se člověka, co udělá řidič po zařazení rychlosti, odpoví vám, že vrátí ruku na volant.

Následující model ukazuje, jak se má chovat neuronová síť při predikci, kterou budu srovnávat s realitou. Řekněme, že $1-10$ jsou stavy řidiče v jednotlivých snímcích videozáznamu. Stav je dán vektorem čísel, ve kterých je zakódovaná poloha řidičova těla. Neuronové síti pak předávám vstup, který je v modelu reprezentován třemi snímky a na výstupu očekávám dva snímky následující. Chceme tedy přesně to, co jsem popsal výše. Síti řeknu: „Řidič právě zařadil, co udělá nyní?“, a očekávám odpověď: „Řidič vrátí ruku na volant.“

stavy řidiče v 10 snímcích:	$[1,2,3,4,5,6,7,8,9,10]$
pro snímek 1:	Nedostatek vstupních hodnot pro predikci
pro snímek 2:	Nedostatek vstupních hodnot pro predikci
pro snímek 3:	$[1,2,3] \Rightarrow [4,5]$
pro snímek 4:	$[2,3,4] \Rightarrow [5,6]$
pro snímek 5:	$[3,4,5] \Rightarrow [6,7]$
	$\vdots$

2. Když si však řidič po zařazení pustí rádio, a pak si např. něco podá, jedná se o anomálii, kterou by měla síť detekovat? Lidský mozek automaticky odpoví negativně (nejedná). V tomto případě bychom od sítě očekávali, že označí chování za normální. Což ale síť pravděpodobně neudělá, protože reaguje na 3 sekundy chování, které právě proběhly - a nejpravděpodobnější se jí bude jevit, že po zařazení vrací řidič ruku na volant.

   Proto jsem experimentoval s druhým přístupem, který je opět blízký lidskému uvažování. Neuronová síť se naučí rozpoznat obvyklé chování tak, že jí dáme tentokrát 3 sekundy záznamu a chceme po ní, aby predikovala tytéž 3 sekundy - tedy „normální“ 3 sekundy.

   Zde se eliminuje nevýhoda předchozího přístupu, protože po síti nechci reakci na uběhnutý sled událostí, ale chci, aby mi rovnou předpověděla 3 sekundy normálního chování, které pak srovnám s realitou. Od takto naučené sítě zjistím, že ruka na řadicí páce je normální, ruka vracející se na volant je normální, ruka putující z řadicí páky směrem k rádiu je normální a podávání pití je také normální. A nezáleží tedy na tom, zda před tím došlo k nějakému „normálnímu“ sledu událostí.

   Opět zvažujeme podobný model. Tentokrát ale síti řeknu: "Řidič právě zařadil, je to nor-

mální? Udělala bys to také?", a očekávám odpověď: "Ano, také bych to udělala."

$$
\begin{aligned}
\text{stavy řidiče v 10 snímcích:} &\quad [1,2,3,4,5,6,7,8,9,10] \\
\text{pro snímek 1:} &\quad \text{Nedostatek vstupních hodnot pro predikci} \\
\text{pro snímek 2:} &\quad \text{Nedostatek vstupních hodnot pro predikci} \\
\text{pro snímek 3:} &\quad [1,2,3] \Rightarrow [1,2,3] \\
\text{pro snímek 4:} &\quad [2,3,4] \Rightarrow [2,3,4] \\
\text{pro snímek 5:} &\quad [3,4,5] \Rightarrow [3,4,5] \\
&\quad \vdots
\end{aligned}
$$

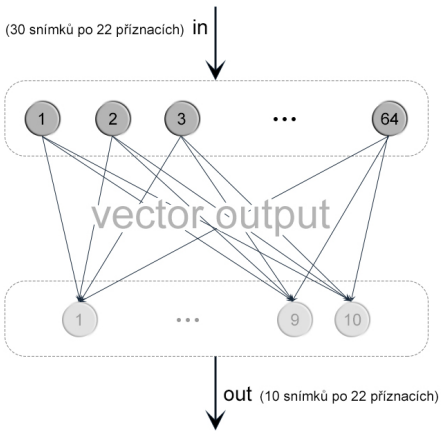
V obou případech je zde zřejmý fakt, že síť není schopna odpovídat prvních $N-1$ snímků, kde N je počet snímků určených pro vstup neuronové sítě, protože nemá dostatek vstupních podnětů. Tyto situace prohlašuji za *normální*.

Jak je vidět, v každém případě je třeba zvolit takový způsob výpočtu sítě (model sítě), který umí předpovídat vícekrokový výstup časové řady [23]. Pro svou práci jsem zvolil dva modely vhodného typu. První, přímočarý, „model vektoru výstupů“ a druhý model typu „enkodér-dekodér“.

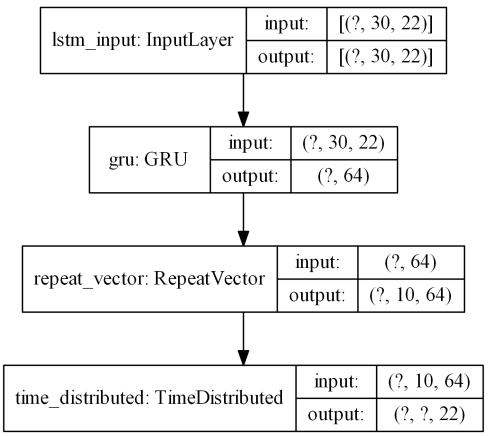
5.9 Použité modely neuronové sítě

5.9.1 Vector Output Model (model vektoru výstupů)

Jedná se o přímočarý model, který je možný implementovat pomocí většiny neuronových sítí vč. LSTM i GRU, které v experimentech používám. Pomocí něj neuronová síť počítá a na výstupu přímo vrací vektor hodnot reprezentující předpověď více časových kroků do budoucnosti.



Obrázek 11: Vizualizace sítě typu Vector Output Model



Obrázek 12: Vector Output Model vygenerovaný pomocí knihovny Keras [4.2.4]

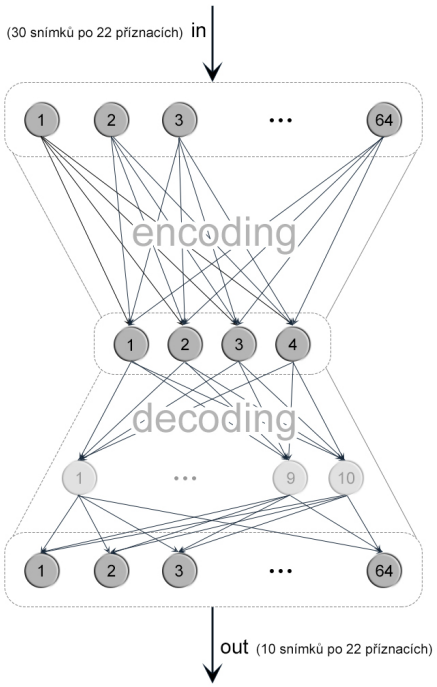
Příklad modelu tvořený v jazyce Python pomocí frameworku TensorFlow [4.2.3] a knihovny Keras [4.2.4]:

```
from keras.models import Sequential
from keras.layers import GRU, RepeatVector, TimeDistributed, Dense

# chci, aby síť pro 30 snímků predikovala 10 následujících snímků
# každý stav řidiče je popsán 22 čísly
input_size, output_size, features_number = 30, 10, 22

model = Sequential()
model.add(
    GRU(64, activation='relu', input_shape=(input_size, features_number))
)
model.add(RepeatVector(output_size))
model.add(TimeDistributed(Dense(features_number)))

model.compile(optimizer='adam', loss='mse')
```

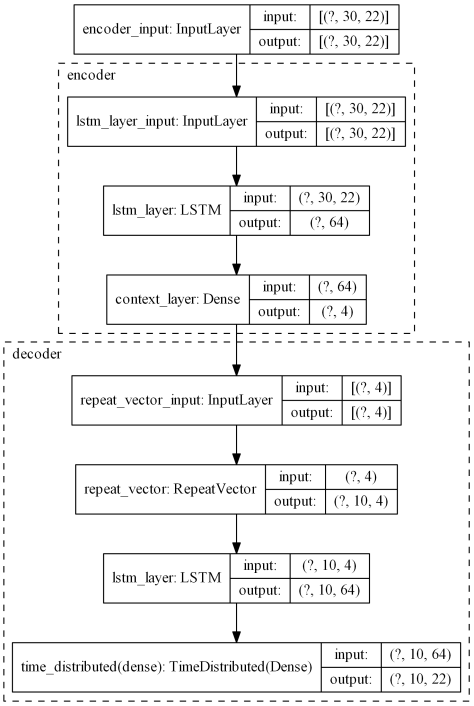
Výpis 2: Vector Output Model RNN

5.9.2 Encoder-Decoder Model (enkodér-dekodér)

Model typu enkodér-dekodér je typickým zástupcem RNN modelu určeného pro řešení tzv. „sequence-to-sequence“ problémů. To znamená takových úloh, které mají na vstupu obecnou posloupnost hodnot a na výstupu vektor hodnot jiné délky a obecně i jiné domény. Model typu enkodér-dekodér, jak již název napovídá, je složeninou dvou různých modelů RNN. Enkodér dostává na vstupu sekvenci určité (proměnné) délky, zpracuje ji a zakóduje (encode) do vrstvy s vektorem fixní délky, typicky nižší kvůli úmyslnému vypuštění nerelevantních informací. Tento vektor se obvykle nazývá „kontextový“, protože obsahuje sémantickou reprezentaci vstupní posloupnosti zvanou kontext. Kontext dále vstupuje do dekodéru, kde je rozkódován (decode) a dekodér vrací vektor hodnot požadované (proměnné) délky na výstup.



Obrázek 13: Vizualizace sítě typu Encoder-Decoder Model



Obrázek 14: Encoder-Decoder Model vygenerovaný pomocí knihovny Keras [4.2.4]

Opět jednoduchý příklad modelu v jazyce Python + TensorFlow a Keras:

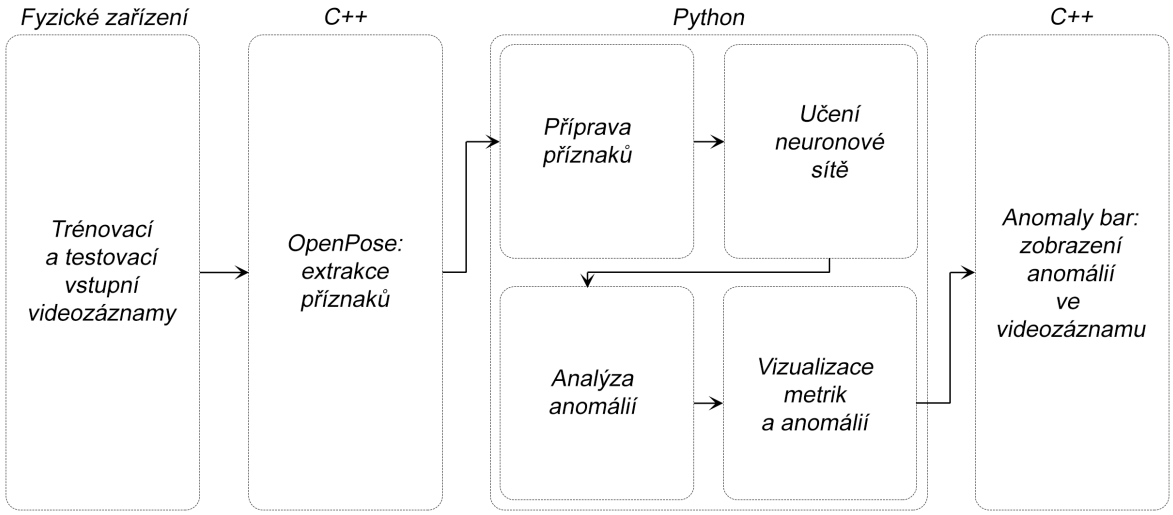
```
from keras.models import Sequential
from keras.layers import LSTM, Dense, RepeatVector, TimeDistributed

# chci, aby síť pro 30 snímků predikovala 10 následujících snímků
# každý stav řidiče je popsán 22 čísly
input_size, output_size, features_number = 30, 10, 22
latent_dimension = 4

# enkodér
encoder = Sequential(name='encoder')
encoder.add(LSTM(64, activation='relu', input_shape=(input_size, features_number)))
encoder.add(Dense(latent_dimension, name='context_layer'))

# dekodér
decoder = Sequential()
decoder.add(RepeatVector(output_size, input_shape=(latent_dimension, )))
decoder.add(LSTM(64, activation='relu', return_sequences=True))
decoder.add(TimeDistributed(Dense(features_number)))
```

```
# enkodér−dekodér
model = Sequential()
model.add(encoder)
model.add(decoder)

model.compile(optimizer='adam', loss='mse')
```

Výpis 3: Encoder-decoder model RNN

6 Experimenty

V této a dalších kapitolách budu podávat výsledky naměřených experimentů. Všechny datové podklady jsem si tvořil sám. Do auta jsem na zpětné zrcátko připevnil web kameru, pomocí které jsem se nahrával za jízdy. Jako zodpovědný řidič jsem všechna tímto způsobem nabytá videa prohlásil za taková, která obsahují „normální“ chování a použil jsem je pro trénování umělé neuronové sítě.



Obrázek 15: Workflow - proces jednoho experimentu od vstupu k vizualizaci detektoru anomálií

V diplomové práci jsem měl ověřit, zda je možné softwarově detekovat anomálie v chování řidiče automobilu. Proto je nedílnou součástí práce i mnou naprogramovaný software. Skládá se ze dvou částí [obr. 15].

C++ [8.1]

Je určená k práci s videem. Extrahuje souřadnice kloubů řidiče pomocí knihovny OpenPose [4.2.2] a vizualizuje detekované anomálie do videozáznamu.

Obrázek 16: Anomálie v původním videu

Obrázek 17: Detekce aplikací - vykreslený tzv. Anomaly bar (ukazatel míry anomálie)

Python [8.2]

Pracuje s extrahovanými daty. Primárně je tato část určená k práci s neuronovými sítěmi využitím platformy TensorFlow [4.2.3] a knihovny Keras [4.2.4]. Data získaná v druhé části aplikace vhodně transformuje a získá z nich příznaky pro neuronové sítě. Výstupem jsou soubory různého typu:

- detekované anomálie
- obrázky s vizualizacemi učícího procesu neuronových sítí (ztrátová funkce a přesnost)
- graf anomálií

Všechny tři vizualizace ukazuji dále v textu ve vyhodnocení experimentů [6.1].

Ve svých experimentech jsem bral v potaz faktory uvedené v následujícím seznamu. Jedná se o různá nastavení určující konfiguraci a stav sítě. Zkoušel jsem i mnohé další. Např. jsem do sítí přidával další vrstvy, měnil jsem počty LSTM/GRU jednotek, zkoušel jsem rozdílné počty vstupních a výstupních vektorů aj. Většina z těchto modifikací, ale nepřinášela žádné zlepšení. Některé úpravy sice zlepšily přesnost predikce sítě, ale pouze v rámci desetin či setin jednotek na úkor mnohem delšího trénovacího času. Jedinou výjimkou bylo zvýšení dimenze latentní vrstvy modelu typu Encoder-Decoder. Tím se ale ztrácel význam Encoderu, tudíž jsem pro pozorování zvolil hodnotu 4, která v několika případech také vykazuje vynikajících výsledků. Než vypíši faktory, které jsem v experimentech testoval (měnil), uvedu konstantní hodnoty, které jsem se rozhodl používat napříč všemi experimenty:

- každá LSTM/GRU vrstva má 64 jednotek
- latentní vrstva modelu Encoder-Decoder má 4 neurony
- počet epoch učení sítě je nastaven na 128 s tím, že jsem aplikoval callback, který učení přeruší, pokud 8 epoch nedošlo k redukci chyby sítě

- všechny sítě jsem trénoval na stejné trénovací sadě o 127031 záznamů, kde 118138 záznamů sloužilo k učení a 7%, tj. 8893 vzorků, k validaci
- pro validaci jsem využíval dvě metriky:
  - *chybová funkce (loss) typu MSE:*
    minimalizace střední kvadratické chyby mezi požadovanými hodnotami a predikcí sítě
  - *přesnost (accuracy):*
    frekvence s jakou předpovězená hodnota odpovídá hodnotě požadované
- pro detekci anomálií jsem u všech experimentů využíval stejné množiny testovacích dat
- kvalitu detekce anomálií na testovaných videozáznamech jsem vyhodnocoval dvěma způsoby
  - *poměr síly signálu a nechtěného šumu (SNR):*
    následující kroky popisují výpočet SNR (čím vyšší výsledek, tím lépe)
    1. ručně jsem anotoval sekvence (tzn. vyznačil jsem / definoval, ve kterých časových intervalech jsou anomálie)
    2. jednotlivé úseky jsem rozšířil na každou stranu o přechodový interval δt, kde nehodnotím, zda detektor rozpoznal abnormální chování či nikoliv
    3. k výpočtu využívám soubor s rozdíly predikce a reality tak, že spočítám průměrnou hodnotu míry anomálií v intervalech, kde se anomálie nachází S (signal) a tuto hodnotu vydělím průměrnou hodnotou míry anomálií v intervalech, kde se anomálie nenachází N (noise) - $SNR = \frac{S}{N}$
  - *úspěšnost:*
    Procentuální vyjádření správně predikovaných dat. Každý predikovaný rámec (snímek) je srovnáván s realitou. Jestliže síť ve srovnávaném snímku správně detekovala anomálii, nebo správně rozhodla, že se jedná o normální chování, úspěšnost roste a naopak.
- 4 nejúspěšnější konfigurace byly testovány na dalších dvou videozáznamech (na jednom z nich byla při řízení natáčena jiná osoba, než na kterou byla síť naučena)

Faktory aplikované na experimenty:

1. přístup k predikci (resp. ke vstupnímu a výstupnímu vektoru příznaků)
   - učení z minulosti (vstup = 30 snímků, predikce = následujících 10 snímků)
   - učení na identitu (vstup = 20 snímků, predikce = stejný vektor jako ten vstupní)
2. typ jednotek RNN sítě

- LSTM
- GRU

3. typ sítě
   - jednosměrná
   - obousměrná
4. příznaky
   - pozice kloubů
   - úhly mezi vybranými spojnicemi kloubů
5. model
   - Vector Output Model
   - Encoder-Decoder Model

Jedná se o 5 faktorů, každý z nich může nabývat dvou hodnot, tzn. celkem 32 zdokumentovaných experimentů.

6.1 Naměřené výsledky

Reálný počet epoch se lišil na skrze experimenty, a zdá se, že obousměrné sítě potřebují k naučení méně epoch. Tento fakt není pravidlem. Velmi odlišná však byla doba učení jednotlivých experimentů. Téměř žádnou roli nehrál typ zvolených příznaků (přestože úhlů bylo 16 a pozic kloubů 22), a ani velikost vstupního a výstupního vektoru. Zajímavým poznatkem je, že dle pozorování se LSTM síť učila kratší dobu než síť typu GRU, a přitom LSTM je složitější struktura.
Z hlediska délky učení byly zásadní tři faktory:

1. LSTM vs. GRU
2. Vector Output Model vs. Encoder-Decoder Model
3. jednosměrná vs. obousměrná síť

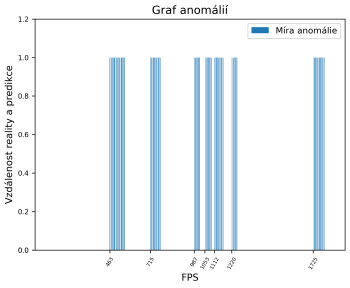
V konkrétních číslech se jednalo o následující hodnoty (počet sekund je na jednu epochu):

- LSTM, jednosměrná, Vector Output Model: **75 sekund**
- LSTM, jednosměrná, Encoder-Decoder Model: **114 sekund**
- GRU, jednosměrná, Vector Output Model: **79 sekund**
- GRU, jednosměrná, Encoder-Decoder Model: **150 sekund**
- LSTM, obousměrná, Vector Output Model: **154 sekund**

- LSTM, obousměrná, Encoder-Decoder Model: **230 sekund**
- GRU, obousměrná, Vector Output Model: **227 sekund**
- GRU, obousměrná, Encoder-Decoder Model: **295 sekund**

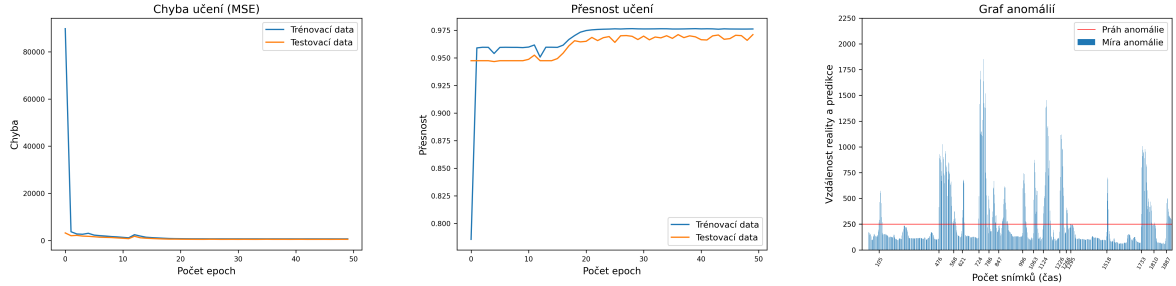
V následujícím přehledu uvádím výsledky všech experimentů. Číslování odpovídá identifikátoru používanému v naprogramované aplikaci [8]. Výsledky jsem koncipoval tak, že obsahují všechny relevantní informace:

- **první řádek** - metriky sledované při učení a vyhodnocování (hodnoty odpovídají naučené síti), vysvětlení viz seznam faktů (konstant) týkajících se všech experimentů:
  - *chyba* - hodnota minimalizované MSE
  - *přesnost* - hodnota naměřená na validační frakci dat
  - *snr* - vypočítaný poměr síly signálu a šumu
  - *úspěšnost* - vypočítaná úspěšnost detekce normálního/abnormálního stavu
- **druhý řádek** - podává informaci o konfiguraci sítě (odpovídá faktorům popsaným výše):
  - typ sítě
  - typ jednotek RNN sítě
  - model
  - příznaky
  - přístup k predikci
- ve třetím řádku jsou vizualizace procesu učení (metrik) a detekce anomálií:
  - ztrátová funkce
  - přesnost učení
  - vizualizace detekce anomálií na testovacích datech (vysoké hodnoty poukazují na vysokou míru anomálie)

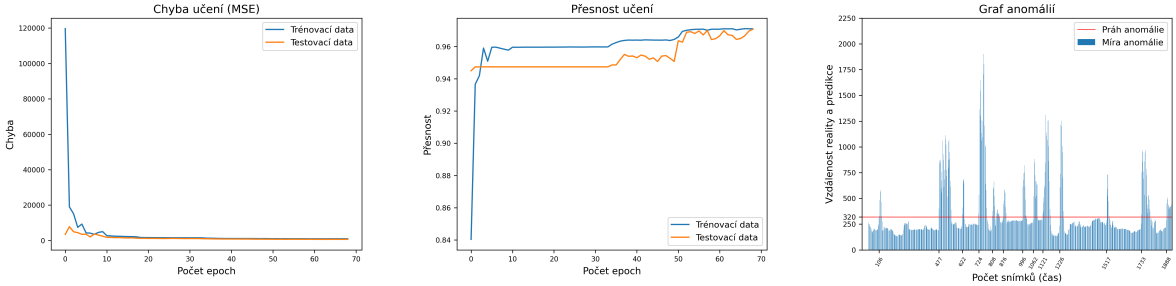


Obrázek 18: Manuálně vytvořené anotace sekvencí anomálií v testovacím datasetu používané pro výpočet SNR a úspěšnosti.

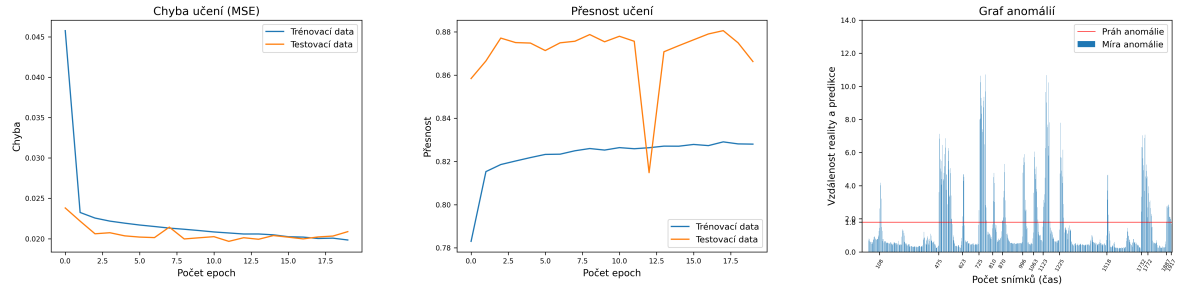
Obrázek 19: Experiment 1
→ chyba = **696.8868**, přesnost = **97.58**%, snr = **4.1285**, úspěšnost = **90.12**%
→ jednosměrná, LSTM, Vector Output Model, pozice kloubů, učení z minulosti



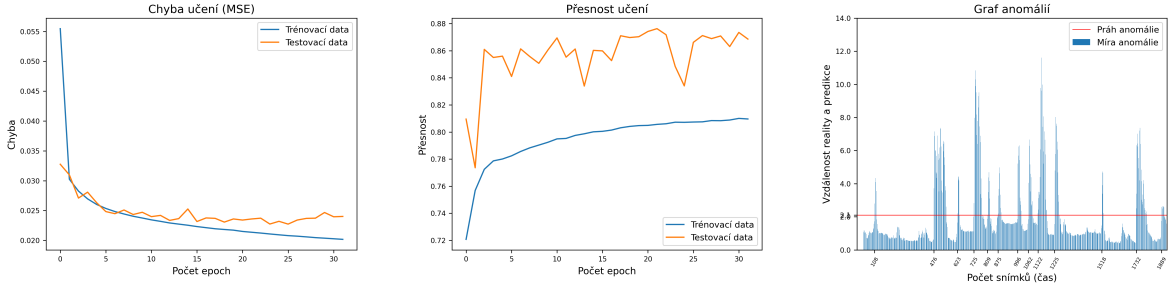
Obrázek 20: Experiment 2
→ chyba = **924.9451**, přesnost = **97.05**%, snr = **2.8137**, úspěšnost = **92.67**%
→ jednosměrná, LSTM, Encoder-Decoder Model, pozice kloubů, učení z minulosti



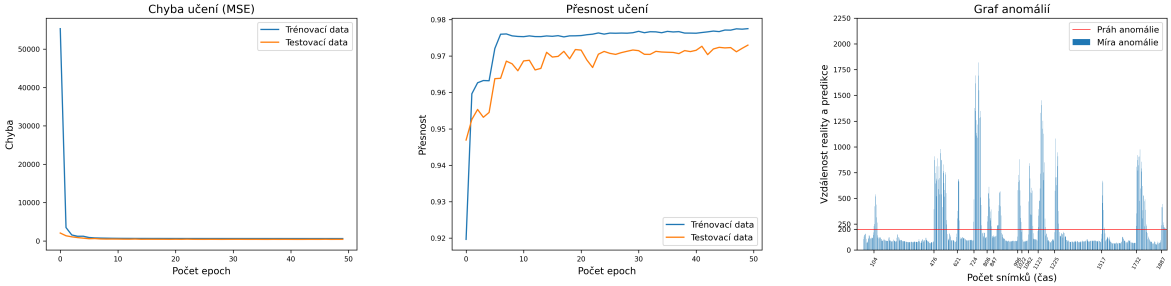
Obrázek 21: Experiment 3
→ chyba = **0.0205**, přesnost = **83.0**%, snr = **5.4696**, úspěšnost = **94.38**%
→ jednosměrná, LSTM, Vector Output Model, úhly mezi klouby, učení z minulosti



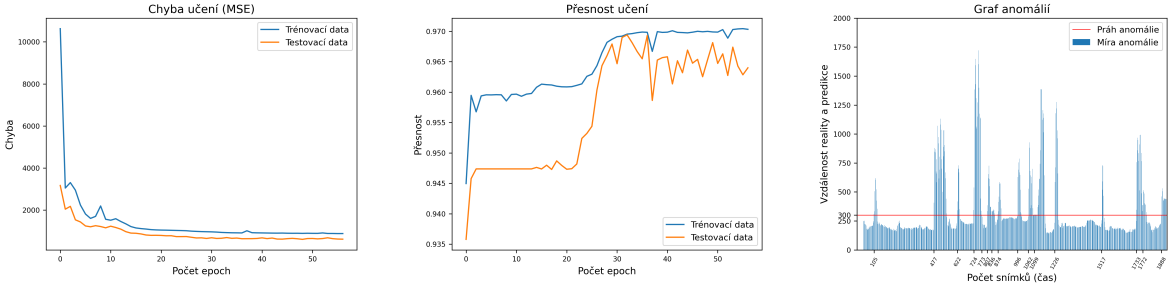
Obrázek 22: Experiment 4
→ chyba = **0.0212**, přesnost = **79.32**%, snr = **4.0549**, úspěšnost = **95.22**%
→ jednosměrná, LSTM, Encoder-Decoder Model, úhly mezi klouby, učení z minulosti



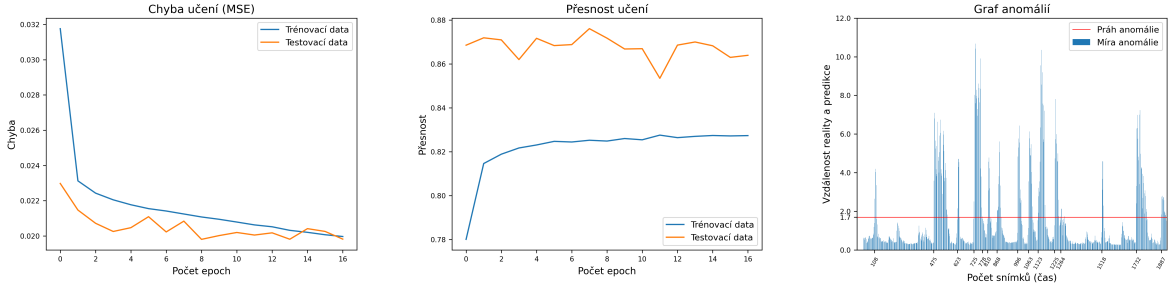
Obrázek 23: Experiment 5
→ chyba = **595.3434**, přesnost = **97.71**%, snr = **4.9714**, úspěšnost = **92.04**%
→ obousměrná, LSTM, Vector Output Model, pozice kloubů, učení z minulosti



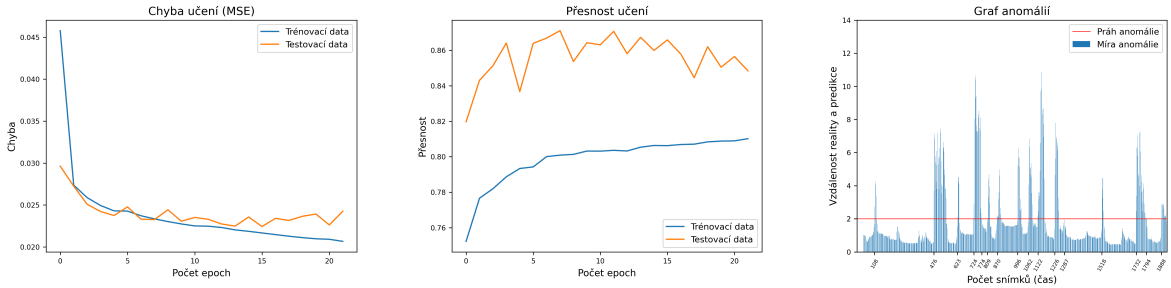
Obrázek 24: Experiment 6
→ chyba = **869.0178**, přesnost = **96.97**%, snr = **2.9442**, úspěšnost = **92.15**%
→ obousměrná, LSTM, Encoder-Decoder Model, pozice kloubů, učení z minulosti



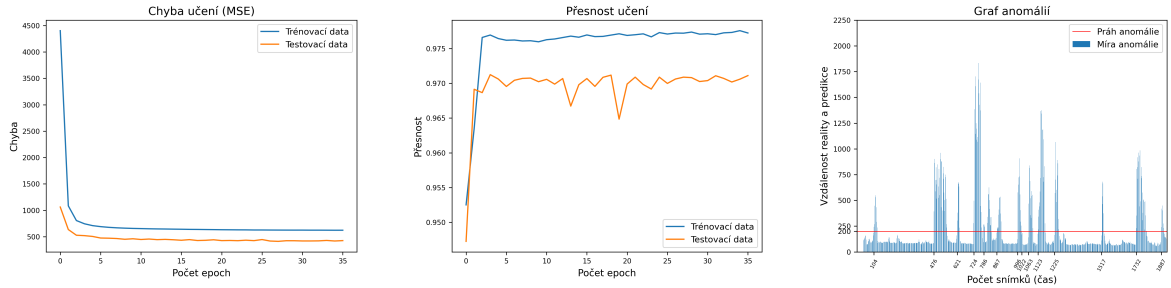
Obrázek 25: Experiment 7
→ chyba = **0.0206**, přesnost = **83.6**%, snr = **5.4623**, úspěšnost = **93.97**%
→ obousměrná, LSTM, Vector Output Model, úhly mezi klouby, učení z minulosti



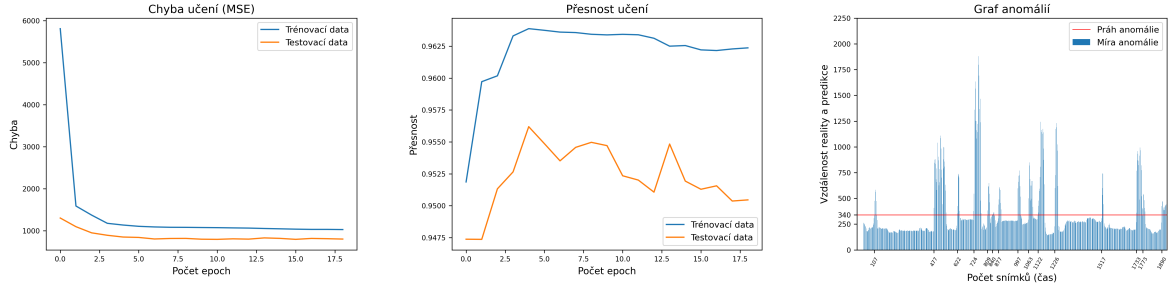
Obrázek 26: Experiment 8
→ chyba = **0.0217**, přesnost = **80.86**%, snr = **4.1386**, úspěšnost = **94.49**%
→ obousměrná, LSTM, Encoder-Decoder Model, úhly mezi klouby, učení z minulosti



Obrázek 27: Experiment 9
→ chyba = **604.7816**, přesnost = **97.77**%, snr = **5.1152**, úspěšnost = **93.24**%
→ jednosměrná, GRU, Vector Output Model, pozice kloubů, učení z minulosti

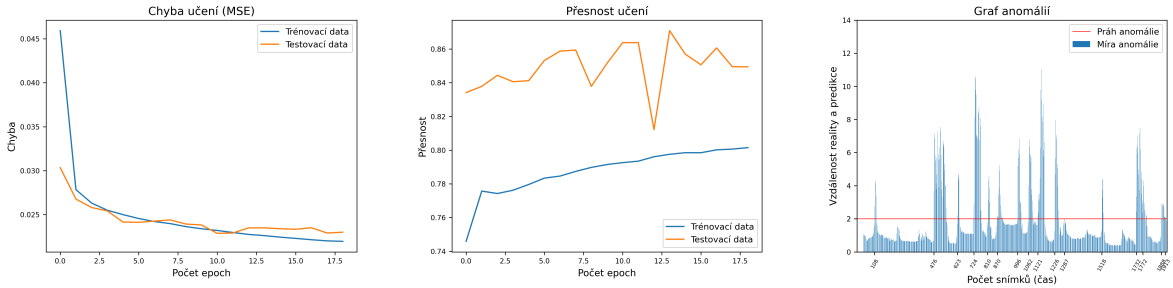


Obrázek 28: Experiment 10
→ chyba = **1043.09**, přesnost = **96.23**%, snr = **2.7237**, úspěšnost = **93.6**%
→ jednosměrná, GRU, Encoder-Decoder Model, pozice kloubů, učení z minulosti

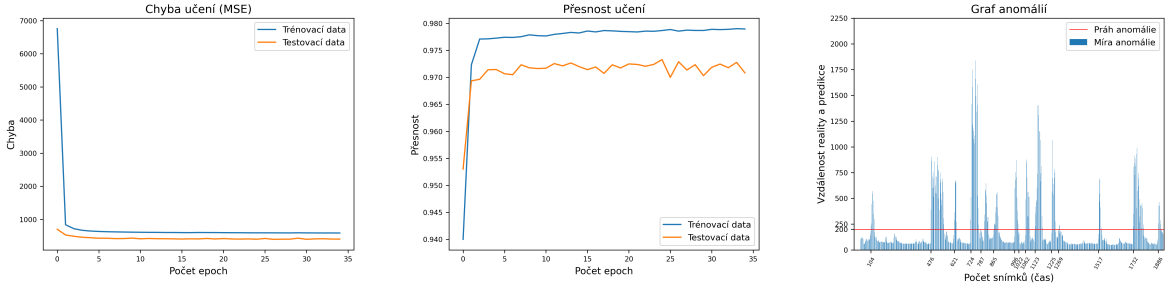


Obrázek 29: Experiment 11
→ chyba = **0.0214**, přesnost = **83.37**%, snr = **5.594**, úspěšnost = **94.23**%
→ jednosměrná, GRU, Vector Output Model, úhly mezi klouby, učení z minulosti

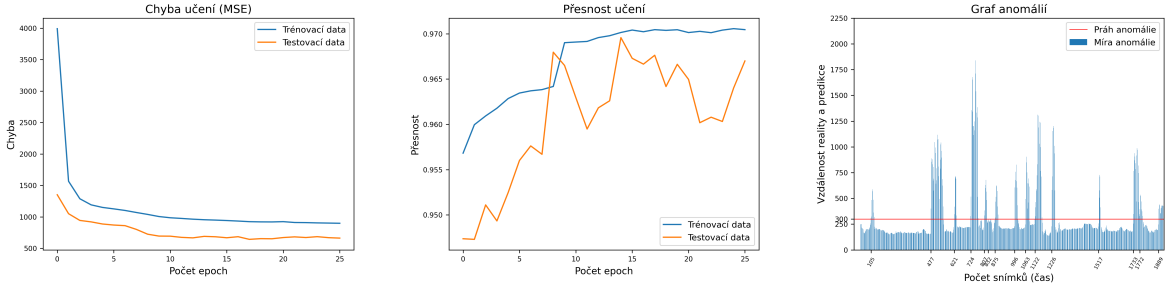
Obrázek 30: Experiment 12
→ chyba = **0.0229**, přesnost = **80.7**%, snr = **4.043**, úspěšnost = **93.97**%
→ jednosměrná, GRU, Encoder-Decoder Model, úhly mezi klouby, učení z minulosti



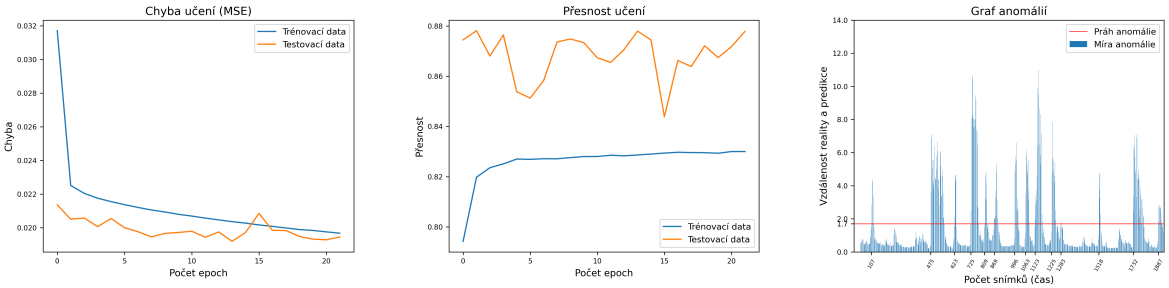
Obrázek 31: Experiment 13
→ chyba = **568.8402**, přesnost = **97.87**%, snr = **5.2632**, úspěšnost = **92.51**%
→ obousměrná, GRU, Vector Output Model, pozice kloubů, učení z minulosti



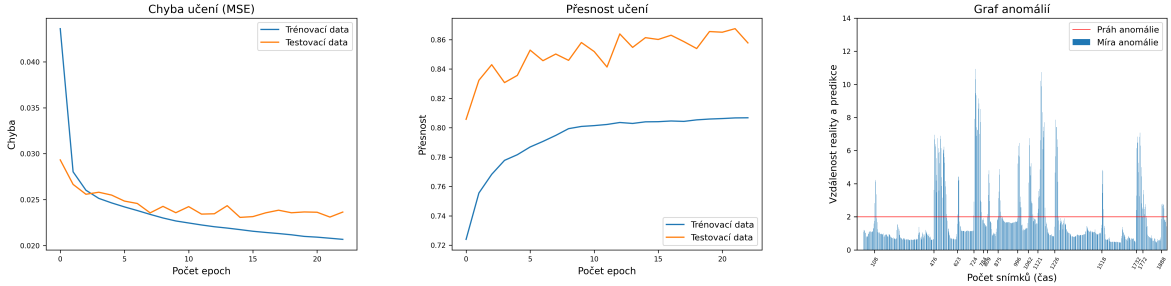
Obrázek 32: Experiment 14
→ chyba = **894.8764**, přesnost = **97.05**%, snr = **3.0593**, úspěšnost = **93.45**%
→ obousměrná, GRU, Encoder-Decoder Model, pozice kloubů, učení z minulosti



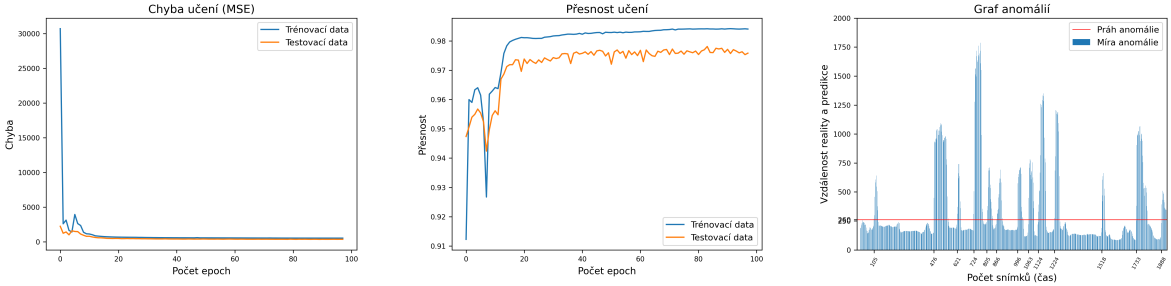
Obrázek 33: Experiment 15
→ chyba = **0.02**, přesnost = **84.02**%, snr = **5.6334**, úspěšnost = **94.33**%
→ obousměrná, GRU, Vector Output Model, úhly mezi klouby, učení z minulosti



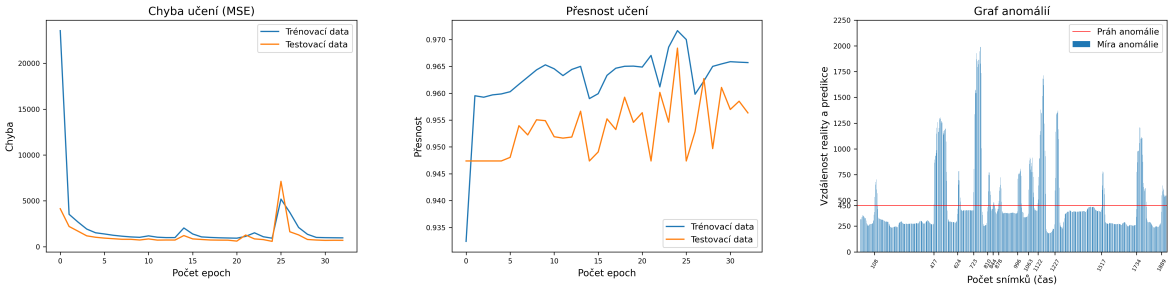
Obrázek 34: Experiment 16
→ chyba = **0.0216**, přesnost = **81.26**%, snr = **3.916**, úspěšnost = **95.27**%
→ obousměrná, GRU, Encoder-Decoder Model, úhly mezi klouby, učení z minulosti



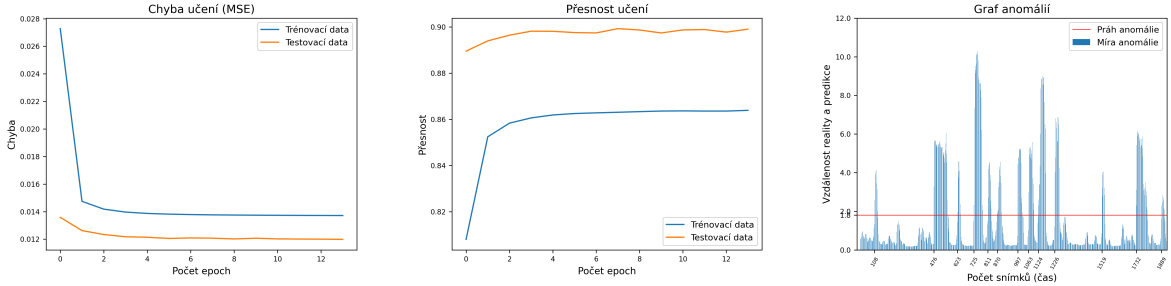
Obrázek 35: Experiment 17
→ chyba = **526.4223**, přesnost = **98.37**%, snr = **4.0555**, úspěšnost = **90.9**%
→ jednosměrná, LSTM, Vector Output Model, pozice kloubů, učení na identitu



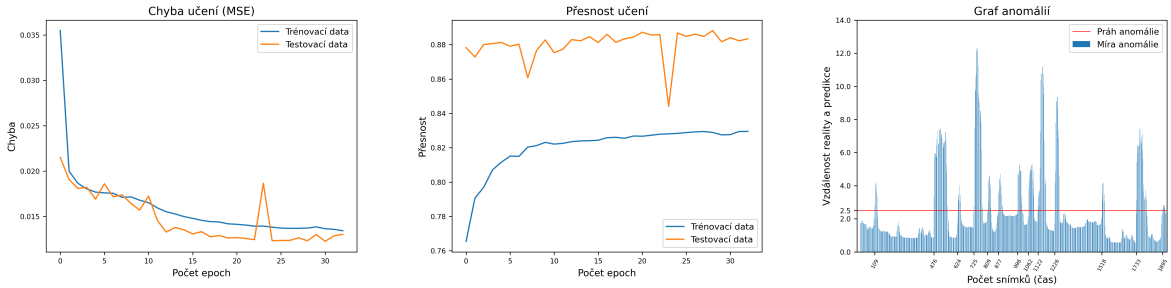
Obrázek 36: Experiment 18
→ chyba = **893.4362**, přesnost = **97.16**%, snr = **2.7991**, úspěšnost = **92.67**%
→ jednosměrná, LSTM, Encoder-Decoder Model, pozice kloubů, učení na identitu



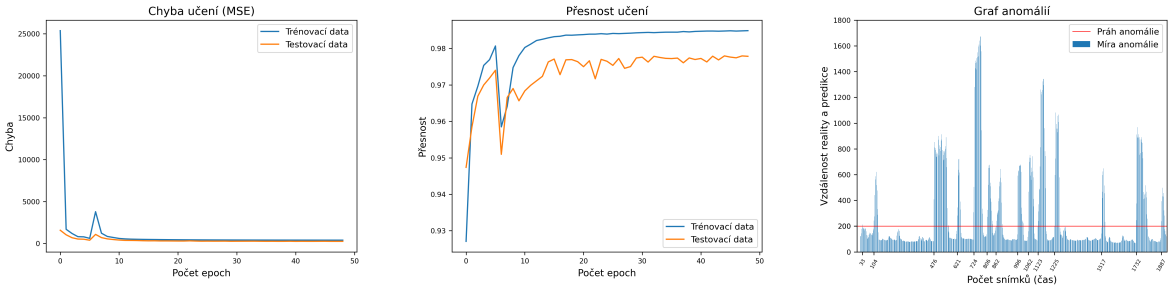
Obrázek 37: Experiment 19
→ chyba = **0.0137**, přesnost = **86.42**%, snr = **6.4052**, úspěšnost = **93.45**%
→ jednosměrná, LSTM, Vector Output Model, úhly mezi klouby, učení na identitu



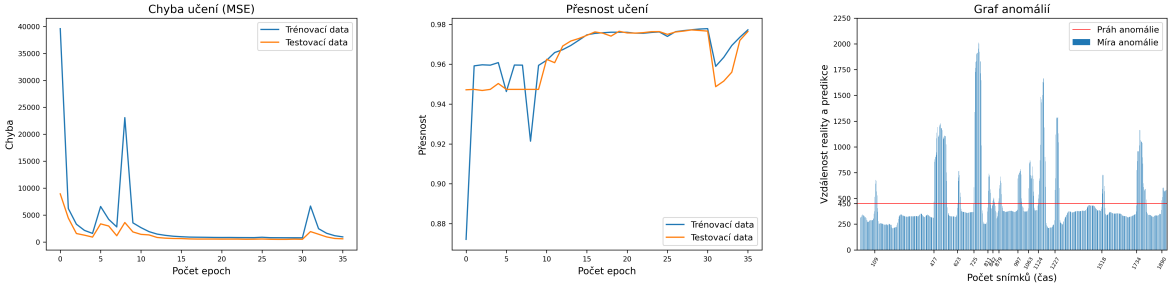
Obrázek 38: Experiment 20
→ chyba = **0.0136**, přesnost = **83.67**%, snr = **3.6973**, úspěšnost = **93.71**%
→ jednosměrná, LSTM, Encoder-Decoder Model, úhly mezi klouby, učení na identitu



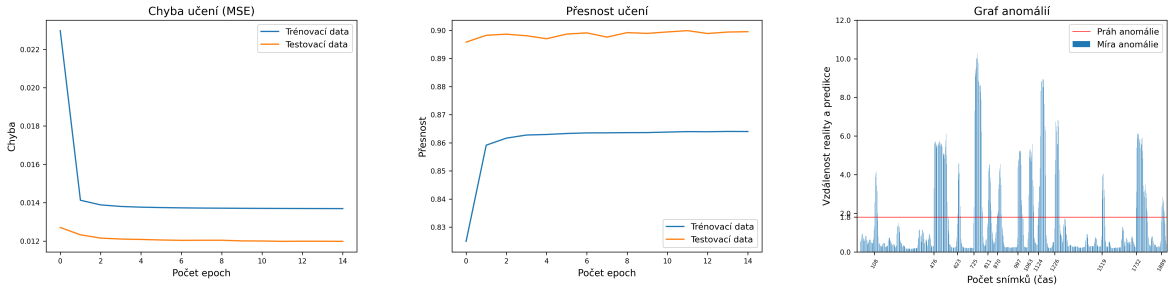
Obrázek 39: Experiment 21
→ chyba = **392.9151**, přesnost = **98.42**%, snr = **5.0923**, úspěšnost = **91.0**%
→ obousměrná, LSTM, Vector Output Model, pozice kloubů, učení na identitu



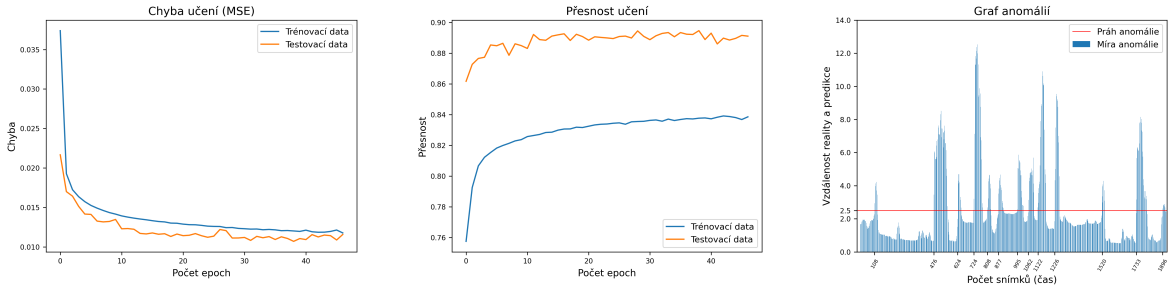
Obrázek 40: Experiment 22
→ chyba = **781.9968**, přesnost = **97.65**%, snr = **2.6108**, úspěšnost = **92.67**%
→ obousměrná, LSTM, Encoder-Decoder Model, pozice kloubů, učení na identitu



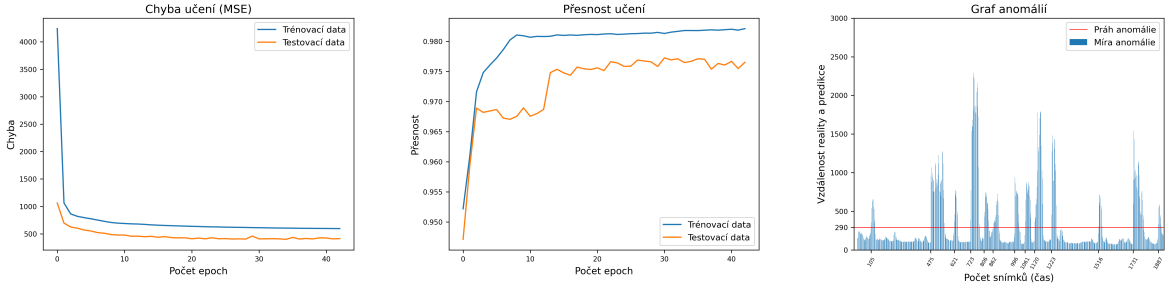
Obrázek 41: Experiment 23
→ chyba = **0.0136**, přesnost = **86.63**%, snr = **6.4885**, úspěšnost = **93.45**%
→ obousměrná, LSTM, Vector Output Model, úhly mezi klouby, učení na identitu



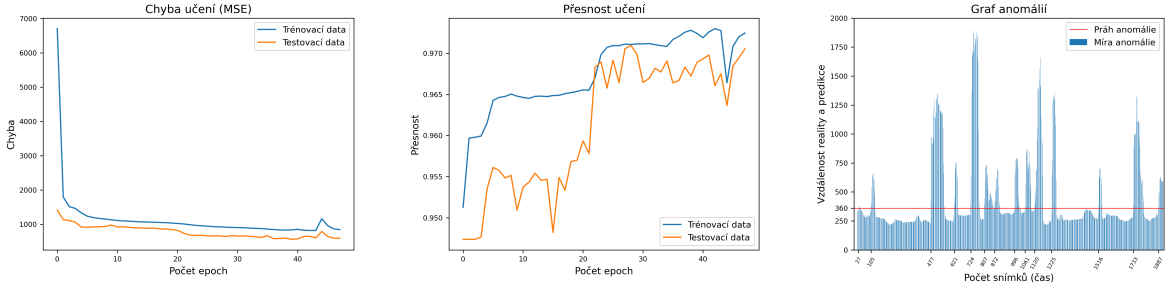
Obrázek 42: Experiment 24
→ chyba = **0.0117**, přesnost = **84.48**%, snr = **3.7509**, úspěšnost = **93.19**%
→ obousměrná, LSTM, Encoder-Decoder Model, úhly mezi klouby, učení na identitu



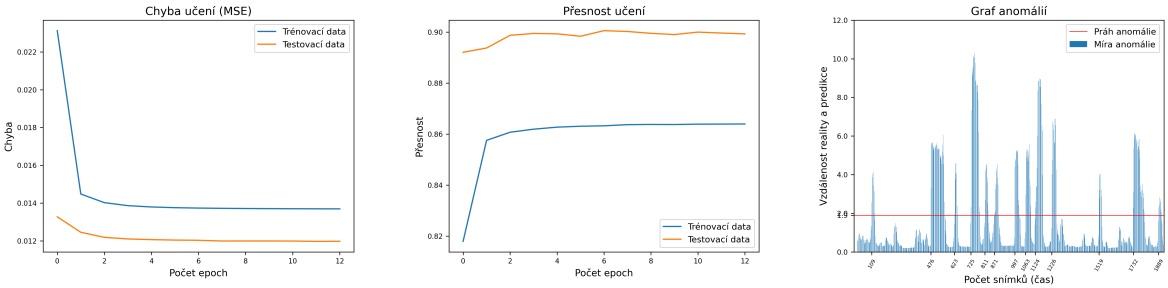
Obrázek 43: Experiment 25
→ chyba = **580.2091**, přesnost = **98.15**%, snr = **5.2927**, úspěšnost = **91.63**%
→ jednosměrná, GRU, Vector Output Model, pozice kloubů, učení na identitu



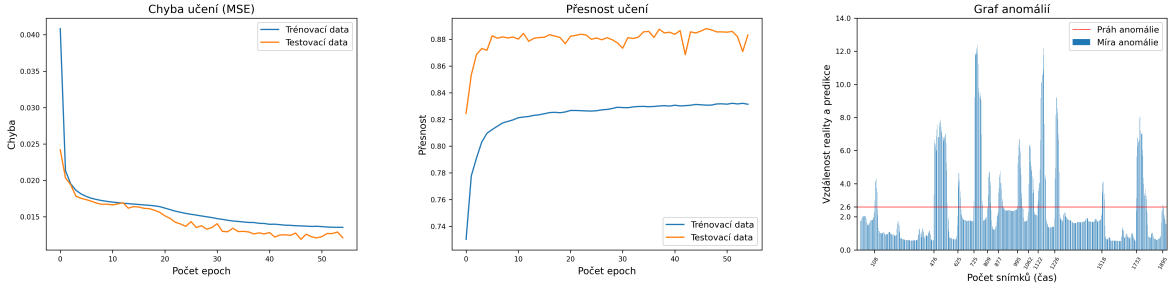
Obrázek 44: Experiment 26
→ chyba = **788.2319**, přesnost = **97.35**%, snr = **2.9809**, úspěšnost = **88.98**%
→ jednosměrná, GRU, Encoder-Decoder Model, pozice kloubů, učení na identitu



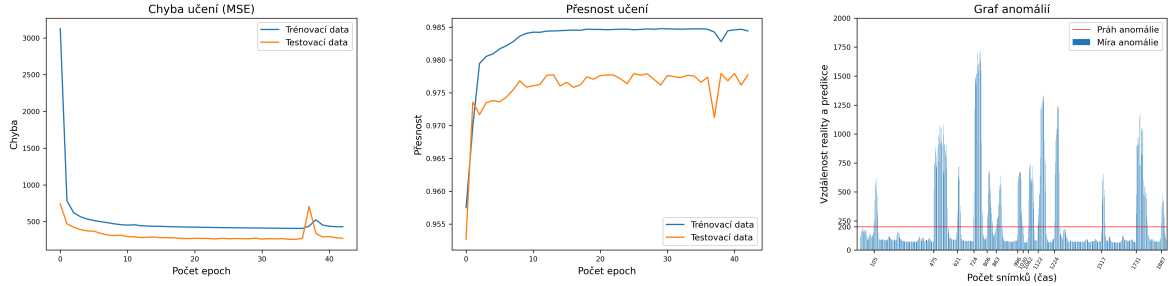
Obrázek 45: Experiment 27
→ chyba = **0.0137**, přesnost = **86.58**%, snr = **6.2831**, úspěšnost = **93.66**%
→ jednosměrná, GRU, Vector Output Model, úhly mezi klouby, učení na identitu



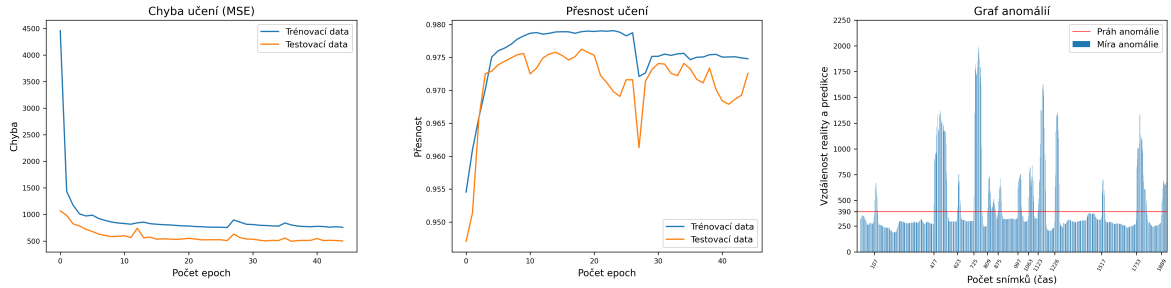
Obrázek 46: Experiment 28
→ chyba = **0.0135**, přesnost = **83.63**%, snr = **3.8343**, úspěšnost = **94.07**%
→ jednosměrná, GRU, Encoder-Decoder Model, úhly mezi klouby, učení na identitu



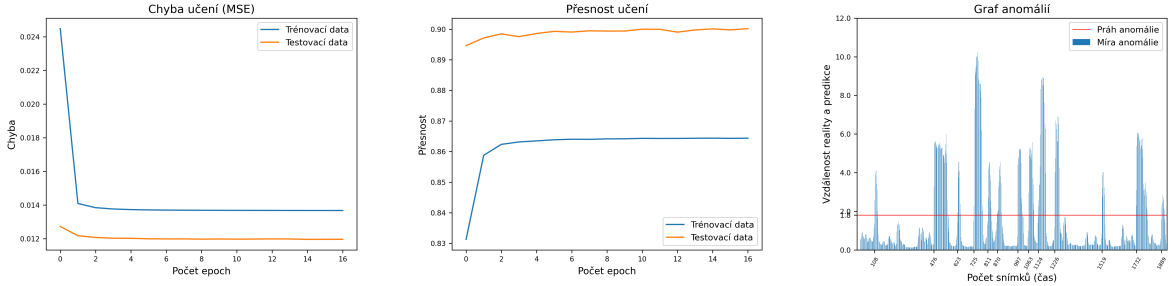
Obrázek 47: Experiment 29
→ chyba = **395.2532**, přesnost = **98.43**%, snr = **5.7698**, úspěšnost = **91.32**%
→ obousměrná, GRU, Vector Output Model, pozice kloubů, učení na identitu



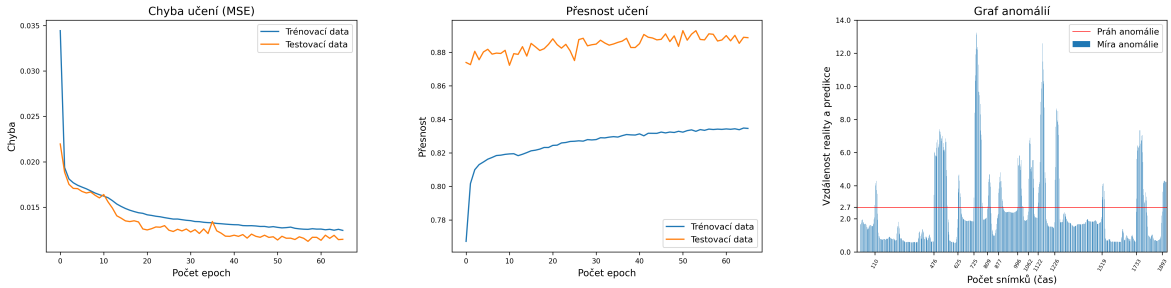
Obrázek 48: Experiment 30
→ chyba = **748.0532**, přesnost = **97.44**%, snr = **2.9624**, úspěšnost = **90.95**%
→ obousměrná, GRU, Encoder-Decoder Model, pozice kloubů, učení na identitu



Obrázek 49: Experiment 31
→ chyba = **0.0136**, přesnost = **86.66**%, snr = **6.5597**, úspěšnost = **93.45**%
→ obousměrná, GRU, Vector Output Model, úhly mezi klouby, učení na identitu



Obrázek 50: Experiment 32
→ chyba = **0.0123**, přesnost = **84.02**%, snr = **3.6919**, úspěšnost = **92.98**%
→ obousměrná, GRU, Encoder-Decoder Model, úhly mezi klouby, učení na identitu



6.2 Nejlepší konfigurace

V této kapitole vyberu čtyři nejlepší konfigurace, které použiji pro experimenty s druhým a třetím datovým souborem. Druhý testovaný videozáznam obsahuje jiné anomálie, než ten předchozí, na kterém jsem testoval všechny konfigurace. A ze třetí nahrávky budu detekovat anomálie u jiné osoby, než na kterou byla síť naučena.

Dle grafů vizualizace detekce anomálií je zjevné, že Encoder-Decoder Model je odolnější vůči šumu. Na tento fakt poukazuje vyšší „základna“ grafů. Jedná se o očekávaný a žádoucí projev, který způsobuje latentní vrstva sítě. Informace ze sítě jsou zakódovány do nižší dimenze. Tím pádem přichází o detailní informace, které by jinak mohly způsobovat určitou míru šumu. Zajímavé je, že z hlediska přesnosti učení normálnímu chování je ve většině případů Encoder-Decoder Model značně horší než Vector Output Model. Ale z hlediska úspěšnosti tomu tak není. Nízké hodnoty přesnosti jsou z části opět způsobeny dimenzionalitou latentní vrstvy. Při zvětšování dimenze by docházelo ke zlepšování výsledků, ale také by síť odstraňovala čím dál méně šumu. I přesto jsem použitím tohoto modelu naměřil některé velmi přívětivé hodnoty.

Velmi mne překvapilo pozorování, že použitím úhlů mezi spojnicemi vybraných kloubů bylo dosaženo mnohem horších výsledků než použitím pouhých pozic kloubů. A to navzdory tomu, že

použitím „selského rozumu“ se mi příznaky popsané úhly jeví logičtější a odolnější vůči vnějším vlivům. Jedná se o další námět k budoucímu rozšiřování práce. Napadá mě několik možností vylepšení:

1. změna spojnic - zde se ale obávám, že to k lepším výsledkům nepovede
2. úhly v trojrozměrném prostoru - snímat řidiče souběžně z více úhlů a využít OpenPose (nebo jinou knihovnu) k získání pozic kloubů v 3-D prostoru [5.7]
3. změna typu nebo konfigurace neuronové sítě

Je obtížné stanovit nejlepší výsledky, protože je třeba brát v úvahu čtyři odlišné metriky. Důležitější by se mohly jevit hodnoty SNR a úspěšnosti, protože hodnotí kvalitu detektoru. Ale jedná se o hodnoty, které se vztahují ke konkrétním testovacím datům. Jelikož jsou testovací data mnohonásobně menšího rozsahu, je přesnost detekce normálního chování stejně užitečným parametrem. Obdobnými úvahami bychom došli i ke stejné důležitosti dalších dvou metrik. Vzal jsem tedy v potaz všechny čtyři metriky s následující prioritou:

1. **přesnost** naučení detekce normálního stavu
2. **úspěšnost** detekce normálního / abnormálního chování
3. **SNR**
4. **MSE** naměřená na naučené NS

V následující tabulce uvádím konfigurace s nejlepšími výsledky. Všechny dosáhly přesnosti nad 97%, úspěšnosti nad 90%, SNR mají vysoké hodnoty a MSE naopak co nejnižší. Tabulka je seřazena dle čísla experimentu:

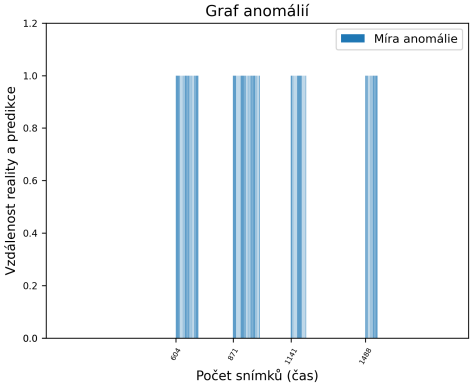
Experiment	Přesnost	Úspěšnost	SNR	MSE
5	97,77	93,24	5,1152	604,7816
13	**97,8700**	**92,51**	**5,2632**	**568,8402**
17	97,71	92,04	4,9714	595,3434
18	**98,15**	**91,63**	**5,2927**	**580,2091**
21	**98,43**	**91,32**	**5,7698**	**395,2532**
22	98,42	91	5,0923	392,9151
26	**98,37**	**90,9**	**4,0555**	**526,4223**
29	97,58	90,12	4,1285	696,8868

Cílem kapitoly bylo vybrat 4 nejlepší experimenty. Jelikož účelem výběru nejlepších experimentů je detekce anomálií dalších datových zdrojů, vybral jsem experimenty napříč různými faktory. Jako nejlepší konfigurace, které budu používat pro další detekci volím experimenty č. 13, 18, 21 a 26 (v tabulce jsou tučně vyznačené).

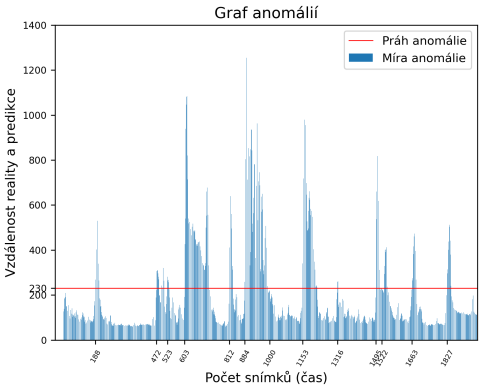
6.3 Druhý dataset

Využitím čtyř zvolených experimentů jsem detekoval i další dva videozáznamy. V druhém z nich jsem řidičem opět já. Tentokrát jsem se zaměřil na „úzké hrdlo“, nad kterým jsem při zkoumání přemýšlel. V záznamu provádím opakovaný pohyb rukou tak, že z řadicí páky zapnu rádio, pak si něco podám z vedlejšího sedadla a teprve poté vracím ruku na volant. Jedná se o normální chování, které by mohlo být problematické pro neuronovou síť predikující budoucí snímky na základě minulosti. Takovýmto experimentem je ze zvolených pouze experiment č. 13.

Poznámka: SNR a úspěšnost u obrázků se vztahuje k druhému datasetu. SNR a úspěšnost v tabulce výše se vztahovaly k sadě testovacích dat používaných při zkoumání všech experimentů.

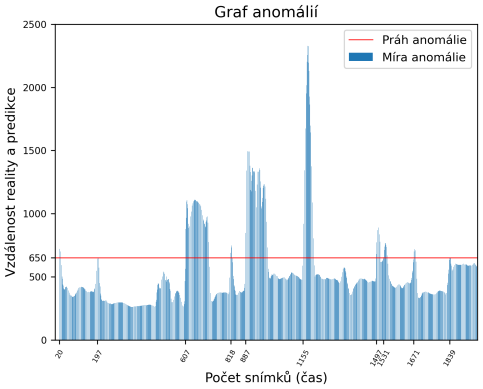


Obrázek 51: Manuálně vytvořené rozsahy anomálií v druhém datasetu.

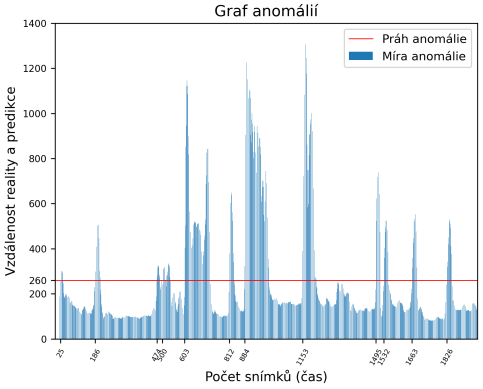


Obrázek 52: Experiment 13, dataset 2
→ snr = **4.032**, úspěšnost = **94.1**%
→ obousměrná, GRU, Vector Output Model, pozice kloubů, učení z minulosti

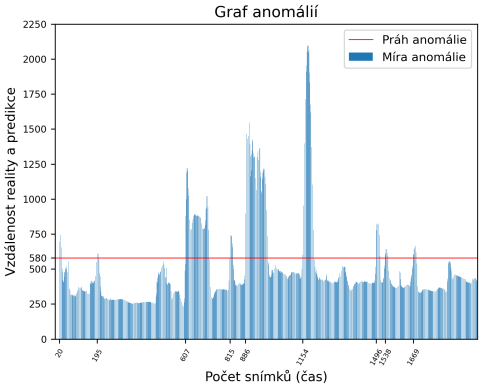
Obrázek 53: Experiment 18, dataset 2
→ snr = **2.4584**, úspěšnost = **98.48**%
→ jednosměrná, LSTM, Encoder-Decoder Model, pozice kloubů, učení na identitu



Obrázek 54: Experiment 21, dataset 2
→ snr = **3.7952**, úspěšnost = **91.9**%
→ obousměrná, LSTM, Vector Output Model, pozice kloubů, učení na identitu



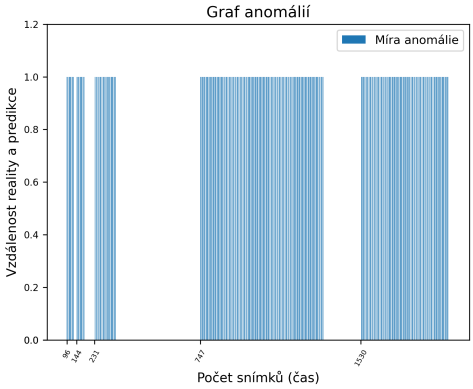
Obrázek 55: Experiment 26, dataset 2
→ snr = **2.5245**, úspěšnost = **97.56**%
→ jednosměrná, GRU, Encoder-Decoder Model, pozice kloubů, učení na identitu



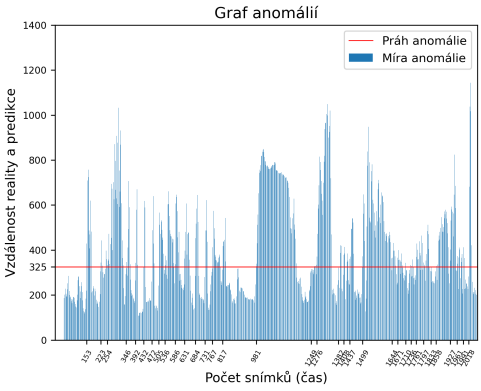
Z obrázků jde vidět, že zkoumané „úzké hrdlo“ nezpůsobuje žádné zhoršení kvality detekce. Experiment č. 13 dosáhl sice druhé nejhorší úspěšnosti, ale nejlepší hodnoty poměru síly signálu a nechtěného šumu. Detekce využitím vybraných experimentů dopadla velmi dobře. Úspěšnosti zůstaly nad 90%, dvě ze čtyř dokonce okolo 98%.

6.4 Detekce anomálií jiné osoby

Třetím testovaným videozáznamem jsem chtěl zjistit, zda je detektor schopný rozpoznávat anomálie i u osoby, na kterou nebyly neuronové sítě naučené. Požádal jsem proto manželku, zda by mi s mým experimentem mohla vypomoci. Moje žena je značně rozdílných rozměrů, než já, a tudíž sedí za volantem v jiné poloze. Podrobil jsem tedy videozáznam testování a s překvapením jsem zjistil, že detektor velmi spolehlivě rozpoznává i abnormální chování jiné osoby. Při zkoumání tohoto experimentu vyvstalo opravdové úzké hrdlo celého detektoru, a tím je volba prahu, který rozděluje hodnoty normálního a abnormálního chování. Jak je vidět na grafech, je zjevné, že má drahá polovička byla pro detektor „abnormální“ po celou dobu - samozřejmě ve srovnání se mnou. Musel jsem proto zvýšit prahovou hodnotu a problém byl vyřešen. Výsledek experimentu č. 26 byl naprosto vynikající a přesáhl hranici 99% úspěšnosti. Volba prahu je opravdovým kandidátem na další zkoumání.

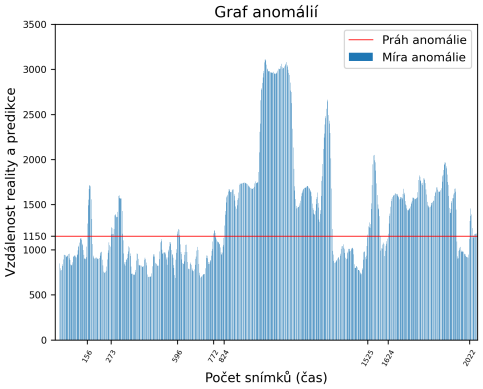


Obrázek 56: Manuálně vytvořené rozsahy anomálií ve třetím datasetu.

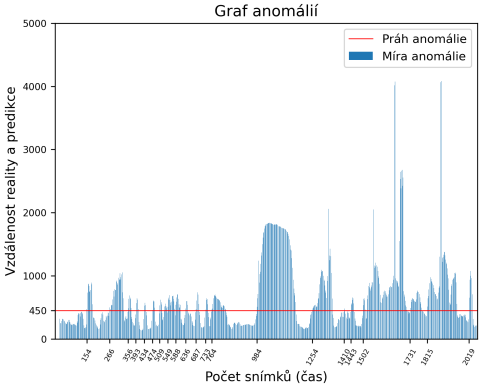


Obrázek 57: Experiment 13, dataset 3
→ snr = **1.4911**, úspěšnost = **87**%
→ obousměrná, GRU, Vector Output Model, pozice kloubů, učení z minulosti

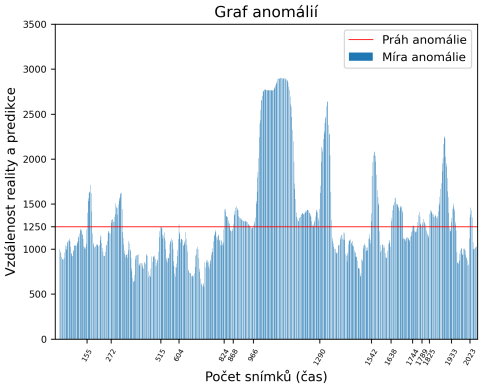
Obrázek 58: Experiment 18, dataset 3
→ snr = **1.896**, úspěšnost = **97.72**%
→ jednosměrná, LSTM, Encoder-Decoder Model, pozice kloubů, učení na identitu



Obrázek 59: Experiment 21, dataset 3
→ snr = **2.2652**, úspěšnost = **89.08**%
→ obousměrná, LSTM, Vector Output Model, pozice kloubů, učení na identitu



Obrázek 60: Experiment 26, dataset 3
→ snr = **1.6748**, úspěšnost = **99.08**%
→ jednosměrná, GRU, Encoder-Decoder Model, pozice kloubů, učení na identitu



7 Vyhodnocení výsledků

V této kapitole shrnu výsledky, kterých jsem dosáhl při zkoumání úlohy detekce anomálií v chování řidiče. Shrnu dříve podrobně popsané informace týkající se hodnocení výsledků. Na stejné testovací datové sadě jsem zkoumal 32 různých experimentů. Nejlepší 4 jsem poté použil na prozkoumání odlišných datasetů. Pozoroval jsem 4 různé metriky:

1. **přesnost** naučení detekce normálního stavu
2. **úspěšnost** detekce normálního / abnormálního chování
3. **SNR**
4. **MSE** naměřená na naučené NS

Kompletní přehled výsledků uvádím v následující tabulce. Z hlediska přesnosti učení neuronových sítí jsem naměřil nejnižší hodnotu 79,32% a nejvyšší 98,43%. Z hlediska kvality detekce byl rozsah úspěšnosti mezi 88,98% a 95,27% a rozsah poměru síly signálu a nechtěného šumu mezi 2,6108 a 6,5597.

Čtyři nejlepší zvolené experimenty dosáhly výborných výsledků i na dalších dvou videozáznamech. Jelikož byl jeden z nich i videozáznam osoby, na kterou nebyl detektor naučený, a vzhledem k faktu, že je zde ještě dostatek prostoru pro další zkoumání a potenciální vylepšování detektoru, jeví se mi výsledky více než uspokojivé a projekt úspěšný.

Přehledová tabulka:

Experiment	Přesnost	Úspěšnost	SNR	MSE
1	96,2300	93,6000	2,7237	1043,0900
2	97,0500	93,4500	3,0593	894,8764
3	83,0000	94,3800	5,4696	0,0205
4	79,3200	95,2200	4,0549	0,0212
5	97,7700	93,2400	5,1152	604,7816
6	97,0500	92,6700	2,8137	924,9451
7	83,6000	93,9700	5,4623	0,0206
8	80,8600	94,4900	4,1386	0,0217
9	97,1600	92,6700	2,7991	893,4362
10	97,6500	92,6700	2,6108	781,9968
11	83,3700	94,2300	5,5940	0,0214
12	80,7000	93,9700	4,0430	0,0229
13	97,8700	92,5100	5,2632	568,8402
14	96,9700	92,1500	2,9442	869,0178
15	84,0200	94,3300	5,6334	0,0200
16	81,2600	95,2700	3,9160	0,0216
17	97,7100	92,0400	4,9714	595,3434
18	98,1500	91,6300	5,2927	580,2091
19	86,4200	93,4500	6,4052	0,0137
20	83,6700	93,7100	3,6973	0,0136
21	98,4300	91,3200	5,7698	395,2532
22	98,4200	91,0000	5,0923	392,9151
23	86,6300	93,4500	6,4885	0,0136
24	84,4800	93,1900	3,7509	0,0117
25	97,4400	90,9500	2,9624	748,0532
26	98,3700	90,9000	4,0555	526,4223
27	86,5800	93,6600	6,2831	0,0137
28	83,6300	94,0700	3,8343	0,0135
29	97,5800	90,1200	4,1285	696,8868
30	97,3500	88,9800	2,9809	788,2319
31	86,6600	93,4500	6,5597	0,0136
32	84,0200	92,9800	3,6919	0,0123

8 Vytvořený software

Software jsem vyvíjel na svém notebooku DELL Precision 7520 s grafickou kartou NVIDIA Quadro M1200. Ovladač grafické karty verze 442.19. Počítač je osazen 32GB operační paměti. Aplikace by však neměla být závislá na hardwaru, možná až na výjimku grafické karty, protože nástroje OpenPose a Tensorflow byly instalovány s podporou CUDA 10.1. Myslím si ale, že pro spuštění aplikace bude podstatná pouze verze těchto nástrojů. Softwarové závislosti uvedu v jednotlivých podkapitolách [8.1, 8.2].

Vyvíjel jsem na operačním systému Windows 10 Enterprise. Jak jsem již uvedl dříve, aplikace se skládá ze dvou částí [obr. 15]. První je určena pro práci s videozáznamem. Je napsaná v jazyce C++ [8.1] a postavená na knihovnách OpenCV a OpenPose. Druhá část, která slouží k práci s neuronovými sítěmi, je napsána v jazyce Python [8.2] a je postavená na platformě TensorFlow. Část aplikace programovanou v C++ jsem implementoval ve vývojovém prostředí Microsoft Visual Studio Community 2019 (verze 16.4.5). A pro vývoj druhé části jsem používal IDE PyCharm 2020.1 (Professional Edition).

Každá část softwaru má nějaké požadavky, které je potřeba splnit, než bude aplikace spouštěna, viz kapitoly [8.1, 8.2]. Obě části se spouští z příkazového řádku s použitím několika argumentů. Bližší informace podávám v následujících podkapitolách [8.1, 8.2].

Ještě, než popíši jednotlivé části, zdokumentuji hierarchii adresářů projektu:

- msc_thesis (kořenový adresář projektu)
  - media (testovaná videa)
  - output (výstupní data z obou částí SW)
    - analysis (výsledky analýzy experimentů)
    - figures (vygenerované grafy experimentů)
    - models (uložené modely natrénovaných NS)
    - openpose (výstupy z první části SW - extrahované klíčové body)
  - pus0024_msc_thesis_anomalies_analysis (naprogramované aplikace)
    - documents (zdrojové kódy diplomové práce - LaTeX)
    - neural_network (zdrojové kódy druhé části aplikace - implementace NS)
    - video_processing (zdrojové kódy první části aplikace - OpenPose + OpenCV)

8.1 Práce s videozáznamem v C++

8.1.1 Softwarové závislosti

- OpenCV
- OpenPose 151 (Překládáno s podporou CUDA 10.1)

Ve vývojovém prostředí MS Visual Studio jsem dodával následující:

Configuration Properties > C/C++ > General > Additional Include Directories

V IDE jsou vložené absolutní cesty. Ve výčtu uvádím pouze části, ze kterých je zjevné, o jakou knihovnu se jedná.

- `openpose\include`
- `opencv\include`
- `caffe\include`
- `caffe\include2`
- `CUDA\v10.1\include`
- `caffe3rdparty\include`

Configuration Properties > Linker > Input > Additional Dependencies

V IDE jsou vložené absolutní cesty ke knihovnám. Ve výčtu uvádím pouze části, ze kterých je zjevné, o jakou knihovnu se jedná.

- `openpose\Release\openpose.lib`
- `CUDA\v10.1\lib\x64\cudart\_static.lib`
- `opencv\x64\vc15\lib\opencv\_world420.lib`
- `caffe3rdparty\lib\gflags.lib`
- `caffe3rdparty\lib\glog.lib`
- `caffe\lib\caffeproto-d.lib`
- `caffe\lib\caffe.lib`
- `caffe\lib\caffeproto.lib`

A dále systémové knihovny:

- `kernel32.lib`
- `user32.lib`
- `gdi32.lib`
- `winspool.lib`
- `shell32.lib`
- `ole32.lib`
- `oleaut32.lib`
- `uuid.lib`
- `comdlg32.lib`
- `advapi32.lib`

8.1.2 Spuštění aplikace

Program se nachází v projektovém adresáři `msc_thesis\pus0024_msc_thesis_anomalies_analysis\video_processing\x64\Release`. Spouští se přes binární soubor `video_processing.exe`. Program bere 4 povinné poziční argumenty. První argument je volba požadované funkce aplikace. V ukázkách níže budu předpokládat, že se nacházíme v adresáři se spustitelným souborem `video_processing.exe`.

Veškeré podrobnosti je možné zjistit přes nápovědu:

```
> video_processing.exe -h
```

V programu jsou dostupné dvě funkce:

1. **Extrakce klíčových bodů** pomocí knihovny OpenPose. Poziční argumenty:
   - `keypoints` - jméno funkce
   - `video` - cesta k videozáznamu, ze kterého budeme chtít extrahovat klíčové body
   - `output_file` - cesta k souboru, kam chceme extrahované body uložit (formát JSON)
   - `openpose_models` - cesta k adresáři s OpenPose modely (u mě `C:\Users\nxf42814\Downloads\Diplomovaprace\openpose\models`)

   Příklad:

   ```
   > video_processing.exe keypoints "C:\path\to\my\video.mp4" "C:\path\to\
     keypoints.json" "C:\path\to\openpose\models"
   ```

2. **Vizualizace anomálií** s uložením do MP4 souboru využitím knihovny OpenCV. Poziční argumenty:
   - `visualizer` - jméno funkce
   - `tested_video` - cesta k videozáznamu, ze kterého jsme získaly klíčové body
   - `dst_file` - cesta k souboru s výsledkem analýzy anomálií (výstup druhé části SW)
   - `output_file` - cesta k souboru pro uložení videa s vizualizací anomálií (formát MP4)

   Příklad:

   ```
   > video_processing.exe visualizer "C:\path\to\my\video.mp4" "C:\path\to
     \anomalies.dst" "C:\path\to\video_anomalies.mp4"
   ```

8.2 Implementace neuronových sítí v jazyce Python

8.2.1 Softwarové závislosti

Všechny SW závislosti jsou uvedené v souboru `msc_thesis\pus0024_msc_thesis_anomalies_analysis\neural_network\requirements.txt`, tudíž je možné je nainstalovat všechny pomocí PIP. Pozor ale na podporu CUDA, se kterou jsem já instaloval TensorFlow. Seznam požadovaných SW knihoven:

- matplotlib==3.2.1
- numpy==1.16.4
- pandas==0.25.2
- tensorflow==2.1.0
- tensorflow-gpu==2.1.0 (instalováno dle návodu na webu TensorFlow s podporou CUDA 10.1)
- pydot==1.4.1
- graphviz==0.14 (je nutné nastavit cestu ke knihovnám do proměnné prostředí PATH)

8.2.2 Spuštění aplikace

Spouštěné skripty (Python moduly) se nachází v projektovém adresáři `msc_thesis\pus0024_msc_thesis_anomalies_analysis\neural_network`. Každá funkce má svůj skript určený ke spuštění. Moduly berou argumenty různých typů. V ukázkách níže budu předpokládat, že se nacházíme v adresáři se skripty.

Veškeré podrobnosti je možné zjistit přes nápovědy:

```
> python run_experiment.py -h
> python visualize_experiments.py -h
> python analyze.py -h
```

Každý skript slouží k jinému účelu:

1. **Analýza experimentů**, skript `run_experiment.py`. Analyzuje předdefinované experimenty popsané v kapitole [6]. Bere dva pojmenované argumenty:
   - `-experiment EXPERIMENT` - číslo experimentu (odpovídají číslům experimentů z kapitoly [6]) nebo hodnota `all`, která zpracuje všechny experimenty.

- `-retrain_prompt_decision RETRAIN_PROMPT_DECISION` - `y` nebo `n`. Může se stát, že soubor s uloženým natrénovaným modelem je poškozen a nejde načíst, v takovém případě bude potřeba jej přetrénovat. V situacích, kdy je trénování potřeba je uživatel vyzván k odpovědi. Pomocí tohoto argumentu je možné odpověď vynutit předem.

Příklad:

```
> python run_experiment.py --experiment 1
> python run_experiment.py --experiment all --retrain_prompt_decision y
```

2. **Generování vizualizace výsledků**, skript `visualize_experiments.py`. Analyzuje předdefinovaný testovací soubor a ukládá vizualizaci výsledků a grafu anomálií. Argumenty nebudu uvádět, protože bere totožné argumenty, jako předchozí bod. Příklad:

```
> python visualize_experiments.py --experiment 1
> python visualize_experiments.py --experiment all --
  retrain_prompt_decision y
```

3. **Analýza vlastního videozáznamu**, skript `run_experiment.py`. Poziční argumenty:

   - `keypoints_path` - cesta k souboru s extrahovanými klíčovými body, které chceme analyzovat (formát JSON)
   - `output_path` - cesta k souboru, do kterého se budou ukládat hodnoty anomálií (obvykle textový soubor s příponou .dst)

   Pojmenované argumenty:

   - `-experiment EXPERIMENT` - číslo experimentu (odpovídají číslům experimentů z kapitoly [6]) nebo hodnota `all`, která zpracuje všechny
   - `-analysis_evaluation ANALYSIS_EVALUATION` - cesta k souboru, kde se nachází ručně vytvořená anotace sekvencí anomálií (data určená k vyhodnocení kvality analýzy)
   - `-plot_analysis PLOT_ANALYSIS` - `y` nebo `n`, v případě hodnoty `y` budou vygenerovány grafy analýzy se stejnou cestou, jako je `output_path`, ale s příponou .png
   - `-threshold THRESHOLD` - vlastní hodnota prahu normálního / abnormálního chování, funguje pouze se zadaným číslem experimentu (ve výchozím stavu se přebírá hodnota z experimentu)

   Příklad:

```
> python analyze.py "C:\path\to\keypoints.json" "C:\path\to\output\
  analysis.dst" --experiment 1 --plot_analysis y --threshold 350 --
  analysis_evaluation "C:\path\to\anotated\anomalies.dst"
```

9 Závěr

Úkolem mé práce bylo zjistit, zda je možné softwarově detekovat anomálie v chování řidiče. Na základě průzkumu, implementace a ověření na systematicky vytvořených třiceti dvou experimentech mohu říci, že detekce anomálií **je možná, a to s dosažením velmi dobrých výsledků**. Jednalo se o práci ve velmi zajímavé doméně, týkající se automobilového průmyslu a moderních technologií. Úloha detekce anomálií v chování je opravdu velmi důležitá, protože se dotýká bezpečnosti silničního provozu. A každý zachráněný život stojí za to.

Při zkoumání jsem používal běžný osobní počítač - notebook. Využíval jsem vlastní datové podklady - videozáznamy. Nedílnou součástí diplomové práce je i software [8], ve kterém jsem naimplementoval detektor, a aplikuji jej na mnou vytvořené experimenty.

Program se skládá ze dvou částí a na spuštění obou částí je nutné splnit určité požadavky (závislosti). Vzhledem k situaci, která mi během vypracovávání nastala - pokažený notebook - jsem aplikaci vyvíjel jak v Linuxu, tak ve Windows, tudíž by měla být po splnění SW závislostí a několika minoritních úpravách spustitelná z obou systémů. Avšak software byl dokončený ve Windows 10 a jedná se proto o doporučenou platformu. Veškeré podrobnosti o vytvořeném softwaru jsem popsal v kapitole [8].

V práci jsem se zaměřil na určitou část detekce anomálií. Jedná se o úlohu, kterou lze řešit mnoha způsoby, a proto si myslím, že je více než vhodná k dalšímu rozšiřování. Během zkoumání jsem objevil určitá „úzká hrdla“ mého přístupu a napadly mě různé modifikace. Snad nejdůležitějším kandidátem pro rozšiřování je volba prahu mezi normálním a abnormálním chováním. Ve své práci jsem hodnotu prahu volil ručně na základě pozorování. Nicméně jsem přesvědčený, že by tuto úlohu mohl zastat nástroj využívající strojového učení či umělé inteligence. Další, velmi důležitou možností úpravy by, vzhledem k mému přesvědčení o užitečnosti využití úhlů mezi spojnicemi kloubů, byla volba příznaků pro popis stavu řidiče. Mým návrhem je využití úhlů mezi spojnicemi určitých kloubů (případně i přímo pozic kloubů) v trojrozměrném prostoru. 3-D prostor by měl podávat přesnější informace o poloze řidiče. Předpokládám, že bude invariantní vůči rušení různého typu. Jako další dobrý námět se mi jeví experimentování s jinými druhy umělých neuronových sítí, např. s generativními typy - GAN, VAE či další. A posledním, nikoliv však jediným možným mnou navrhovaným rozšiřováním by byla kombinace s detekcemi jiného typu, např. s detekcí míry stresu podle obličeje a další.

Výše popsané možnosti rozšiřování jsou z hlediska softwarového. Velmi podstatné bude také rozšíření ve formě využití hardwaru. Jednak by bylo velmi žádoucí více se zabývat přenesením výpočtů na GPU (např. CUDA od společnosti Nvidia), protože by mohlo dojít (nejen) k rapidnímu zrychlení učení sítí. A za druhé, jelikož se jedná o úlohu, která se zabývá analýzou dat v automobilu, bude nutné řešení vyzkoušet i na nějakém vestavěném (embedded) hardwaru, na kterém výpočet reálně poběží.

Během práce jsem poznal několik domén zahrnujících jak teorii, tak poznatky o praktické aplikaci metod pro zpracování obrazu a pro využití strojového učení a umělé inteligence. Jsem

vděčný za to, že jsem úlohu mohl prozkoumat. Práce mi byla velkým přínosem a věřím, že její poznatky poslouží jako podklad k budoucímu rozšiřování tohoto zajímavého úkolu.

Literatura

1. NARAYANAN, Sandeep Nair; MITTAL, Sudip; JOSHI, Anupam. OBD_SecureAlert: An Anomaly Detection System for Vehicles. *Conference: 2016 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2016-05. Dostupné z DOI: 10.1109/SMARTCOMP.2016.7501710.
2. SILVA, Nuno; SOARES, João; SANTOS, Maribel Yasmina; RODRIGUES, Helena. Anomaly Detection in Roads with a Data Mining Approach. *Procedia Computer Science*. 2017-12, roč. 121, s. 415–422. Dostupné z DOI: 10.1016/j.procs.2017.11.056.
3. ED-DOUGHMI, younes; IDRISSI, Najlae. Driver Fatigue Detection using Recurrent Neural Networks. *Conference: the 2nd International Conference*. 2019-03. Dostupné z DOI: 10.1145/3320326.3320376.
4. GHAZAL, Mohammed; HAEYEH, Yasmine Abu; MOHAMMAD, Abdelrahman; GHAZAL, Sara. Embedded Fatigue Detection Using Convolutional Neural Networks with Mobile Integration. *Conference: 2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. 2018-08. Dostupné z DOI: 10.1109/W-FiCloud.2018.00026.
5. ZHANG, Haotian. *A Study on Human Pose Data Anomaly Detection*. San Diego, 2019. Dipl. UC San Diego. Získáno z https://escholarship.org/uc/item/7dn5j316.
6. GATT, T.; SEYCHELL, D.; DINGLI, A. Detecting human abnormal behaviour through a video generated model. In: *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*. 2019, s. 264–270.
7. PERLICH, Claudia. Learning Curves in Machine Learning. 2011-01. Dostupné z DOI: 10.1007/978-0-387-30164-8_452.
8. VIRTANEN, Pauli; GOMMERS, Ralf; OLIPHANT, Travis E.; HABERLAND, Matt; REDDY, Tyler; COURNAPEAU, David; BUROVSKI, Evgeni; PETERSON, Pearu; WECKESSER, Warren; BRIGHT, Jonathan; VAN DER WALT, Stéfan J.; BRETT, Matthew; WILSON, Joshua; JARROD MILLMAN, K.; MAYOROV, Nikolay; NELSON, Andrew R. J.; JONES, Eric; KERN, Robert; LARSON, Eric; CAREY, CJ; POLAT, İlhan; FENG, Yu; MOORE, Eric W.; VAND ERPLAS, Jake; LAXALDE, Denis; PERKTOLD, Josef; CIMRMAN, Robert; HENRIKSEN, Ian; QUINTERO, E. A.; HARRIS, Charles R; ARCHIBALD, Anne M.; RIBEIRO, Antônio H.; PEDREGOSA, Fabian; VAN MULBREGT, Paul; CONTRIBUTORS, SciPy 1. 0. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. 2020. Dostupné z DOI: https://doi.org/10.1038/s41592-019-0686-2.
9. BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*. 2000.

10. CAO, Zhe; HIDALGO, Gines; SIMON, Tomas; WEI, Shih-En; SHEIKH, Yaser. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In: *arXiv preprint arXiv:1812.08008*. 2018.

11. CAO, Zhe; HIDALGO, Gines; SIMON, Tomas; WEI, Shih-En; SHEIKH, Yaser. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. 2018-12, s. 14.

12. MARTÍN ABADI; ASHISH AGARWAL; PAUL BARHAM; EUGENE BREVDO; ZHIFENG CHEN; CRAIG CITRO; GREG S. CORRADO; ANDY DAVIS; JEFFREY DEAN; MATTHIEU DEVIN; SANJAY GHEMAWAT; IAN GOODFELLOW; ANDREW HARP; GEOFFREY IRVING; MICHAEL ISARD; JIA, Yangqing; RAFAL JOZEFOWICZ; LUKASZ KAISER; MANJUNATH KUDLUR; JOSH LEVENBERG; DANDELION MANÉ; RAJAT MONGA; SHERRY MOORE; DEREK MURRAY; CHRIS OLAH; MIKE SCHUSTER; JONATHON SHLENS; BENOIT STEINER; ILYA SUTSKEVER; KUNAL TALWAR; PAUL TUCKER; VINCENT VANHOUCKE; VIJAY VASUDEVAN; FERNANDA VIÉGAS; ORIOL VINYALS; PETE WARDEN; MARTIN WATTENBERG; MARTIN WICKE; YUAN YU; XIAOQIANG ZHENG. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Dostupné také z: `https://www.tensorflow.org/`. Software available from tensorflow.org.

13. CHOLLET, François et al. *Keras* [`https://keras.io`]. 2015.

14. VONDRÁK, Ivo. *Umělá inteligence a neuronové sítě: Určeno pro posl. 4. roč. FEI* [Ostrava: VŠB-Technická univerzita]. 1994. ISBN 80-7078-259-5.

15. OLAH, Christopher. *Understanding LSTM Networks* [online (Colah's blog)]. 2015-08. Dostupné z: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

16. PILÁT, Martin. *Neuronové sítě - RBF sítě a rekurentní sítě* [online (Webové stránky Mgr. Martin Piláta, Ph.D. - Katedra teoretické informatiky a matematické logiky, MFF Univerzity Karlovy v Praze)]. [N.d.]. Dostupné z: https://martinpilat.com/cs/prirodou-inspirovane-algoritmy/neuronove-site-rbf-site-rekurentni-site/.

17. HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-Term Memory. *Neural Computation*. 1997, roč. 9, č. 8, s. 1735–1780. Dostupné z DOI: `10.1162/neco.1997.9.8.1735`.

18. GERS, Felix; SCHMIDHUBER, Jürgen; CUMMINS, Fred. Learning to Forget: Continual Prediction with LSTM. *Neural computation*. 2000-10, roč. 12, s. 2451–71. Dostupné z DOI: `10.1162/089976600300015015`.

19. KOSTADINOV, Simeon. *Understanding GRU Networks* [online (server Towards Data Science)]. 2017-12. Dostupné z: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be.

20. SCHUSTER, Mike; PALIWAL, Kuldip. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*. 1997-12, roč. 45, s. 2673–2681. Dostupné z DOI: `10.1109/78.650093`.

21. LEE, Ceshine. *Understanding Bidirectional RNN in PyTorch* [online (server Towards Data Science)]. 2017-11. Dostupné z: https://towardsdatascience.com/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66.

22. OLAH, Christopher. *Neural Networks, Types, and Functional Programming* [online (Colah's blog)]. 2015-03. Dostupné z: http://colah.github.io/posts/2015-09-NN-Types-FP/.

23. BROWNLEE, Jason. *How to Develop LSTM Models for Time Series Forecasting* [online (server Machine Learning Mastery)]. 2018-11. Dostupné z: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/.